arXiv:2303.06488v1 [cs.DS] 11 Mar 2023

具有距离相关成本的二分搜索

CALVIN LENG 和 DAVID KEMPE

摘要。我们提出一种搜索问题，它推广了直线上的典型二分搜索场景。与二分搜索的设定类似，目标由对手在直线上选定；算法进行查询后，会获知查询结果是正确、过高还是过低。与二分搜索设定（即成本为常数函数的特殊情形）不同，查询的成本是查询点与正确答案之间距离的单调非递减函数；对于过高和过低的查询，还可以采用不同的函数。算法的目标是以最小总成本确定由对手选定的目标。需要注意的是，算法甚至要到最后揭示目标时，才知道自己付出了多少成本。这种抽象涵盖了许多自然场景：决策者通过设定某个数值（例如商品价格、带宽、税率、药物剂量等）进行试验，而所选数值离最优值越远，成本或遗憾就越大。

首先，我们证明，对于任意对称成本函数（即过度估计与估计不足的幅度相同时，产生的成本相同），标准二分搜索算法是一个 4-近似算法。

接着，我们证明，当成本函数是距离的有界次数多项式（包括非对称情形）时，可以通过动态规划求得该问题的最优解；这依赖于对先前查询的合并成本进行仔细编码（请注意，这些成本只有到未来才会揭晓）。随后，我们将问题推广到在树上寻找节点的情形：此时，对查询的反馈是目标所在的树上方向，而成本随查询到目标的树上距离增加。借助 Berendsohn 和 Kozma 提出的 k-割搜索树框架，以及我们针对直线情形提出的思路，当成本函数为有界次数多项式时，我们给出了一个多项式时间近似方案。

1. 引言

二分搜索是最基础的算法之一，其应用远远超出计算机科学的范畴。即使是儿童，通常也能在很小的时候通过“数字猜谜”游戏掌握二分搜索：猜测后得到的反馈只有猜对、猜得太高或猜得太低。在不对正确答案作出额外分布假设的情况下，二分搜索能够使猜出正确数字所需的最坏情况猜测次数最少。

将这一见解应用于现实场景：当需要在只能获得“过高”或“过低”这类反馈的情况下猜测某个未知量时，二分搜索能够最快地收敛。此类应用不胜枚举，例如确定合适的药物剂量、商品定价、最大网络带宽、恒温器的设定值、税率等。考察这些例子可以发现，找到正确数值所需的时间固然重要，但猜测值与正确答案相差甚远也会带来成本。药物剂量严重错误可能对健康造成更大危害；商品价格定得过低会损失更多潜在收入；房间太冷或太热，舒适度会低于温度仅略有偏差的情况，等等。

这表明，可以对标准二分搜索模型作出一种自然的改进：为查询赋予非均匀成本。[1] 尽管已有大量文献（见第1.4 节）研究非均匀查询成本，但这些成本大多与查询本身相关，且不取决于正确答案。在恒温器的例子中，这相当于将恒温器设定为 58 华氏度与 68 华氏度的成本不同，而不论理想温度是多少。虽然这种查询成本取决于查询本身的模型有许多不同的应用，但正如上文所述，成本取决于查询与正确答案之间距离的模型，至少同样自然。

有两项研究探讨了这样一类在线优化问题：所产生的成本同时取决于查询和理想设定。Karp 等人 [23] 研究网络中的拥塞控制。在这里，目标是确定发送数据时的理想带宽。以更高带宽发送会因丢包而产生成本，而以较低带宽发送则会产生机会成本，其数值等于理想带宽与所选带宽之差。Kleinberg 和 Leighton[25, 第 2] 节研究了与同一个单个买家反复交易时的单品定价。在这里，价格设得过高会导致买家完全不购买，而价格设得过低则会产生机会成本，其数值等于买家的估值与所选价格之差。

这一设定中的一个难点如下：算法在终止并观察到正确答案之前，并不知道自己已经产生了多少成本。这使得该设定有别于许多优化问题。

1.1. 基本设定。我们考虑以下基本设定，详见第2节。算法需要识别一个对抗性选择的数字 t，它来自 $\{1,\dots,n\}$。为此，它会反复查询数字 q_i，这些数字来自 $\{1,\dots,n\}$。作为回应，它会得知 $q_i = t$（此时过程终止）、 $q_i < t$ 或 $q_i > t$。在第 i 轮进行查询 q_i 时，算法会产生成本 $f(q_i, t)$。其中， f 是事先提供给算法的已知函数；但是，算法直到最后得知 t 时，才能计算已经产生的成本。由于 t 是由对抗者选择的，算法的目标是使所有 t 选择下的最大总成本 $\max_t \sum_i f(q_i, t)$ 最小。

1.2. 直线上的结果。 首先，在第 3 节中，我们证明在线上，二分搜索是一个常数因子近似算法。

定理1。在线上搜索的设定中，假设成本函数单调且对称[2] （即，$f(q,t) = h(|q-t|)$ 仅取决于 t 和 q之间的距离，而不取决于 $t < q$ 还是 $t > q$，并且 h单调非递减）。那么，始终查询剩余可行集中点的标准二分搜索算法，是一个4-近似算法。

我们强调，这一结果适用于任意距离的单调函数，其涵盖的函数类别比我们的其他结果更广。此外，在第 3 节中，我们证明存在一个常数 $c > 1$，使得二分搜索无法提供 c-近似保证，即使在线性代价函数的情况下也是如此。确定二分搜索的紧确近似保证，是一个值得未来研究的方向。

我们关于直线的第二个主要结果在第 4 节证明：当成本函数是一个可能不对称的低次多项式时，可以在多项式时间内找到最优搜索策略。

[1]标准模型中，每次错误查询的成本为 1。[2]对称性假设对于保证成立至关重要。例如，如果 $q < t$ 产生的成本（实质上）为 0，而 $q > t$ 总是会产生很高的成本，那么线性搜索的表现就会远优于二分搜索。

定理 2. 在线上搜索的情形下，假设成本函数是单调的、可能不对称，并且是一个常数次数多项式。换言之， $f(q,t) = h(q-t)$，其中当 $x < 0$时， $h(x) = h^-(x)$；当 $x > 0$时，$h(x) = h^+(x)$，并且 h^-, h^+是度为 p的非递减多项式。于是，存在一个运行时间为 $O(n^{p^2+3p+3})$ 的算法，可以找到使最大总 incurred 成本最小化的搜索策略。

证明定理的关键洞见 2 在于，先前查询历史中的所有相关信息都可以进行足够简洁的编码，从而支持动态规划。该编码具体利用了成本函数是常数次数多项式这一事实，可以将其视为类似摘要的东西，不过这与通常使用摘要的流式处理场景不同。

成本仅取决于 t 与 之间的距离 q 这一事实 有一定限制；特别是，这意味着定理 2 不适用于重复报价定价场景中的遗憾 [25]（在该场景下，当 $q > t$ 时，遗憾为常数 t），也不适用于拥塞控制场景 [23]（在该场景下，当 $q > t$ 时，机会成本有时也被假设为常数 t）。因此，我们将以下推广扩展到 q 和t的有界次数多项式；其证明见第 5节。为获得多项式运行时间，我们还需要假设成本函数的系数 也被某个关于 n的多项式所界定。

定理 3. 在线上搜索的设定中，假设成本函数 f具有如下形式： $f(q,t) = h(q,t)$，其中 [3] 是一个二元多项式 h，其最高次数为 p，且系数的绝对值不超过 $O(n^s)$，其中 s为某个常数。那么，存在一种运行时间为 $O(n^{3+(p+1)\cdot(p+2s+2)/2})$ 的算法，能够找到一种搜索策略，使其在面对对手时的总 incurred 成本最小。

1.3. 向树的推广。另一种直接的理解方式是将二分搜索视为由路径上的节点构成，而路径本身是无向树的一种特殊情形。于是，对于一次查询 q_i，算法会获知目标节点 t位于以 q_i 的子节点为根的哪棵子树中。事实上，对于标准搜索成本模型（目标是最小化查询次数的最大值），已有大量研究将线上搜索的结果推广到在树中搜索未知节点 [1, 6, 8, 11, 13, 33, 35, 21]。

在这种情况下，算法给定一棵包含 n 个节点的无向树 T，以及一个成本函数 h。对手从树中选择一个节点 t。算法执行一系列查询 q_1、q_2、.....。对于一次查询 q_i，除非 $q_i = t$（此时过程终止），算法会得知 q_i 的哪些邻居是树中严格更接近 t 的唯一邻居。算法进行此查询所产生的成本为 $f(q_i,t)$。当 f 不受限制时，即使 T 是蜘蛛图，该问题仍为 NP 难；这可由加权顶点查询场景直接平凡归约得到（查询节点 q 的成本为 w_q），只需对所有 t 令 $f(q,t) = w_q$ 即可。加权顶点查询场景的 NP 难性已在 [9] 中得到证明。因此，我们将重点讨论满足 $f(q,t) = h(d_T(q,t))$ 的情况，其中 $d_T(q,t)$ 表示 T 中 t 和 q 之间唯一路径的长度，且 h 是一个非递减函数。与之前一样，在目标 t 揭示之前，算法无法观察到所产生的成本。在这种场景下，将针对路径所采用的思路加以推广，即可得到一个多项式时间近似方案（PTAS）[4]:

定理 4. 在具有 n 个节点的树 T 上进行搜索的情形中，假设成本函数 h 单调且为常数次数多项式。换言之， $f(q,t) = h(d_T(q,t))$，其中 h 是次数为 p 的非递减多项式。那么，对于每个 $\epsilon > 0$，都存在一种运行时间为 $O(n^{2+(p^2/2+3p/2+2)\cdot(1+2/\epsilon)})$ 的算法，可以找到一种搜索策略，使其最大总 incurred 成本与最优值之比不超过 $(1+\epsilon)$。

该算法基于与路径情形相同的先前查询历史简洁编码。此外，为避免需要考虑的子树数量呈指数级增长，它仅在 [3] 中定义的 k-割搜索树上进行搜索：这些搜索树的可行集仅由这样的子树组成，即每棵子树与其补集之间至多由 k 条边分隔。对于常数 k，此类子树的数量仅为关于 n 的多项式，并且我们证明，仅基于此类 k-割树的最佳搜索策略，是整体最佳搜索策略的 $(1+2/k)$-近似。

1.4. 相关工作。与我们的工作最为密切相关的是上述 Karp 等人 [23] 以及 Klein- berg 和 Leighton [25]的论文。两者都关注分别出现在拥塞控制和单件商品定价背景下的特定目标函数。在这两种情况下，选择的值过低都会导致机会成本，即可用带宽与所选带宽之差，或买方估值与

[3]同样，我们允许根据 $q < t$ 或 $q > t$ 的情况使用不同的函数 h^-、h^+。[4]目前我们尚不清楚，受限问题是否属于 NP 难问题，无法求得最优解。

成交价格之差。当取值过高时，则会考虑不同的成本：对于物品定价和“严苛”版本的拥塞控制，卖方／发送方将一无所获，因此成本就是真实值。在较为“温和”的拥塞控制版本中，成本为 $\frac{A}{B}\cdot(q-t)$，其中 A、B 为某些整数。[23] 和 [25] 的主要目标是研究所产生的成本／遗憾的渐近行为（在 [25] 中，还会考虑关于买方的不同假设）。不过，作为研究的一部分，Kleinberg 和 Leighton [25] 展示了在反复与同一买方交易的情形下渐近最优的策略；Karp 等人 [23] 则明确给出了上述目标函数（即一次多项式）的动态规划。因此，定理 2 和定理 3 可视为将 Karp 等人 [23] 的部分工作推广到更广泛的成本函数类别。[23] 中的分析给出的运行时间为 $O(n^4)$（对于严苛情形）和 $O(n^4\cdot(A+B))$（对于温和情形）。其对 n 的依赖优于我们定理中声称的结果，但这主要是因为我们的分析经过了简化，且略非最优；我们优先考虑分析的简洁性，而非紧致的指数。请注意，在定理 2 中，我们避免了伪多项式因子（$A+B$）——它出现在 Karp 等人 [23] 的分析中，这是因为他们采用了与定理 3 中类似的摘要。

文献中，对二分搜索所依据的搜索框架已有不少推广。最早的一种推广是寻找偏序集 P 的一个目标理想。一般而言，这个问题已知是 NP 难的[5], , 因此人们研究了特定类型的偏序集和平均情形模型 [5, 11, 30, 31, 32] 。我们尤其关注偏序集具有类树结构的情况。Dereniowski 在 [11] 首先研究了这一模型，并指出，类树偏序集上的偏序集搜索问题等价于树上的搜索问题：通过边查询e，揭示包含目标的连通分量 $T\setminus e$。

结合 Lam 和 Yue [29], 关于最优边排序的早期研究，这一等价关系为树上的边搜索问题和顶点搜索问题都带来了线性时间算法。将研究限制在树（或其他更简单的图）上至关重要，因为 Lam 和 Yue [28] 还证明，一般图上的边排序问题和顶点排序问题均为 NP 难。此后，树上的顶点排序问题和边排序问题的多项式时间算法被重新发现：Ben-Asher、Farchi 和 Newman 最初在 [1], 给出了一个运行时间为 $O(n^4)$ 的算法，Onak 和 Parys [35], 将其改进至 $O(n^3)$，最后 Mozes、Onak 和 Weimann [33] 使用与早期边排序研究相似的方法，将运行时间降至线性时间。

关于路径 [19,26, 34] 和树 [7, 9, 13, 21]上该问题的加权变体与平均情形变体，也已有大量研究。特别是，加权边查询问题可以轻易归约为加权节点查询变体 [7], , 并且已知，即使在直径至多为 4 的树、度至多为 16 [21], 的树以及蜘蛛树 [9] 上，加权边查询问题仍是 NP完全的。Cicalese 等人 [7], 首次提出了近似算法，给出了一个 $O(\log\ n/\log\log\log\ n)$ 近似算法。Cicalese 等人随后在 [9], 改进了次对数项，之后 Dereniowski 等人 [13] 将其改进为一个 $O(\sqrt{\log n})$ 近似算法和一个拟多项式时间近似方案。

在推导树上的多项式时间近似方案时，我们关键地借鉴了 k-割树及其分析。它们源于一系列研究，旨在将动态二叉搜索树（BST）的最优性性质推广到这样一种情形：底层搜索空间是树而非路径。这里的目标是设计树上的动态搜索树，使其在一系列搜索中的运行时间渐近逼近离线设定下最优动态搜索树的运行时间。实现常数竞争比的性质称为动态最优性；这一目标引起了广泛关注，但即使对于二叉搜索树也尚未实现。这一研究方向催生了各种数据结构，例如伸展树 [36], , 它是首个实现 $O(\log n)$ 竞争比的数据结构；以及 Tango 树 [10], , 它们是首个实现已知最佳竞争比 $O(\log\log n)$ 的数据结构。Heeringa、Iordan 和 Theran [20] 首次将这一问题推广到类树偏序；他们实现了 $O(w)$ 的竞争比，其中 w 是该偏序的宽度。随后，Bose 等人于 [4] 将这一问题推广到树上：他们改编了 Tango 树中的思想，得到了相同的 $O(\log\log n)$ 竞争比。特别是，他们引入了 Steiner 闭合搜索树的概念。这类搜索树能够将二叉搜索树（BST）所利用的路径的某些优良性质，推广到树上的搜索树（STT）。Berendsohn 和 Kozma [3] 随后将这一概念推广为 k-割搜索树；这使他们得以将伸展树推广到 STT 的情形。

2. 预备知识

我们用 $\mathbb{N}$ 表示非负整数。

我们用 $T=(V,E)$ 表示底层的（简单无向）树，它有 $n=|V|$ 个顶点。对于节点 $v\in V$，我们用 $T\setminus\{v\}$ 表示从 T 中移除顶点 v 后得到的森林。u 和 v 在 T 中的距离记为 $d_T(u,v)$。对于子集 $S\subseteq V$，我们记 $\partial_T(S)=\{u\in V\setminus S\,|\,(u,v)\in E$，其中某个 $v\in S\}$ 表示 S 在 V 中的边界。当上下文含义明确时，我们有时会在记号中省略 T。

当 T 特指直线，即我们考虑“经典”搜索设定时，我们将 $V = \{1, \ldots, n\}$ 视为 $E = \{(j, j+1) \mid j = 1、\ldots\ldots、n-1\}$，并将 $d(u, v) = |u - v|$。

2.1. 查询模型。现在我们正式定义搜索模型。我们关注这样的确定性算法：通过显式查询目标（即使可以根据先前的查询推断出目标），唯一确定由对手选择的目标顶点 $t \in T$。算法事先知道 T。在每一轮 i中，算法查询一个节点 $q_i \in T$。作为回应，它会得知 $q_i = t$，或者得知 q_i 的哪个邻居比 t 更接近 q_i；也就是说，得知哪个连通分量包含 t。当算法查询 t时，过程终止。在任何时刻，与所有查询的回答相一致的节点集合称为可行集。

查询 q_i 的成本为 $f_T(q_i, t)$，其中 $f_T(t, t) = 0$，也就是说，查询正确时不产生成本。对于一般树 T上的大多数结果，我们假设此成本仅取决于 q_i 与 t之间的距离，即其形式为 $f_T(q_i, t) = h(d_T(q_i, t))$，其中 h 是非递减的非负函数。当 T特指直线时，“猜测过低”和“猜测过高”有明确含义，因此我们考虑一种更一般的模型，其中成本函数可根据猜测过低还是过高而有所不同：$f(q_i, t) = h(q_i - t)$。在这种情况下，我们假设 h 在负参数上非递增，在正参数上非递减，且非负；事实上，对于定理 3，我们考虑进一步的推广：$f(q_i, t)$可以是关于 q_i 和 t 的任意单调有界次数二元多项式，且系数有界。

给定目标 t，查询集合 Q 的成本就是成本 $c(Q) = \sum_{q \in Q} f(q, t)$ 之和。我们只关注确定性算法 $\mathcal{A}$，这类算法不会查询顶点 v，因为根据先前的回答可以推断出 $v \neq t$；我们称这类算法为规范算法。在路径/树上进行搜索时，总存在一种最优的规范算法 [5]；请注意，这对任意图的推广并不成立。规范算法 $\mathcal{A}$ 至多在 n 次查询后终止；并且由于它们是确定性的，查询序列 $\langle q_1, q_2, \ldots \rangle =: Q(\mathcal{A}, t)$ 完全由目标顶点决定。给定一棵树和成本函数（记号中省略），算法所产生的最坏情况代价定义如下：

$$C(\mathcal{A}) = \max_{t \in V} c(Q(\mathcal{A}, t)).$$

目标是设计高效算法 $\mathcal{A}$，以（精确或近似地）最小化 $C(\mathcal{A})$。

2.2. 另一种视角：搜索树。自适应算法可以自然地视为一棵搜索树，用以编码该算法的决策。我们正式定义搜索树如下：

定义1（树上的搜索树）。树 T上的搜索树（STT）是一个有根标记树 $\mathcal{T}$，满足以下性质：节点 $u \in \mathcal{T}$的标签 $\lambda(u)$ 是 T的节点；若 $\mathcal{T}$的根 r标记为 $\lambda(r)$，则 $\mathcal{T} \setminus \{r\}$ 的每棵有根子树都是 $T \setminus \{\lambda(r)\}$的一个不同连通分量对应的 STT；此外，$r$ 在 $\mathcal{T}$中的度等于 $\lambda(r)$ 在 T中的度。

注意，这一定义意味着 $\mathcal{T}$的每个叶节点 u 都必须是单节点对应的 STT；否则，无论选择哪个节点作为 $\lambda(u)$，其度都至少为 1，这意味着 u 必须至少有一个子节点。此外，标签 λ 定义了 $\mathcal{T}$的节点与 T的节点之间的双射。

定义 2。设 $\mathcal{T}$是一个 STT。对于 $\mathcal{T}$的每个节点 u，我们定义

$$\phi_{\mathcal{T}}(u) = \{\lambda(x) \mid x \text{ is a descendant of } u \text{ or } x = u\}.$$

我们称 $\phi_{\mathcal{T}}(u)$为与节点 u 相关联的可行集（原因很快就会显而易见）。

The在 T 上的 STT 与在 T上的确定性搜索策略 $\mathcal{A}$ 之间存在自然对应关系。

(1) 对于每棵根节点标记为 $\lambda(r)$ 的 STT $\mathcal{T}$，相应的搜索策略会首先查询 $\lambda(r)$。如果查询响应确认 $\lambda(r)$是目标，则搜索终止。否则，响应会揭示 $T \setminus \{\lambda(r)$ 中包含 t 的连通分量 $S\}$，算法随后递归地将该变换应用于 $\mathcal{T}$ 中对应的子树。通过对查询序列中的查询次数进行归纳，容易证明，在搜索的任意时刻（对应于节点 u），可行节点集合都等于 $\phi_{\mathcal{T}}(u)$。

[5]考虑对顶点 q_i 进行一次查询，且已知它不是目标顶点。那么，之前必定进行过一次查询q_j，其中 $j < i$ 排除了 $q_i = t$的可能性。对q_j 的响应必定表明了 $t \in S$，其中 S 是q_j的一个特定子树，且不包含 q_i。对 q_i 的响应只会表明 t 位于 q_i 中包含q_j的子树 S' 内，并且 $S' \supseteq S$。（如果$q_i = q_j$，则两者可能相等。）因此，对q_i 的响应不会向算法揭示任何新信息。

(2) 反过来，考虑任意一个规范算法 $\mathcal{A}$。它的第一次查询 q_1 将是 STT 的根 r 上的顶点 $\lambda(r) = q_1$。对于除 $q_1 = t$ 以外的每种回答，响应都会揭示 $T \setminus \{q_1\}$ 中包含 t 的连通分量 S。由于该算法是规范的，之后的查询不会查询 S 之外的任何顶点。因此，第一次查询后，该算法对于连通分量 S而言仍是规范的。根据归纳法，该算法剩余部分的执行对应于一棵 STT。于是，通过将与 $T \setminus \{q_1\}$ 的连通分量 S 对应的每棵子树作为 r 的子树，即可得到 $\mathcal{T}$。同样，通过归纳法容易证明，在搜索的任意时刻（对应于节点 u），可行节点集合都等于 $\phi_{\mathcal{T}}(u)$。

对于任意顶点 $u \in \mathcal{T}$，我们用 $P_{\mathcal{T}}(u)$ 表示树 $\mathcal{T}$ 中从根到 u 的路径上的顶点集合，其中包括 r 和 u。于是，对于任意目标节点 t，与 $\mathcal{T}$ 对应的搜索策略的成本可写为

$$C_{\mathcal{T}}(t) = \sum_{u \in P_{\mathcal{T}}(t)} f(\lambda(u), t),$$

沿着搜索树中所有顶点的可行节点集中都包含 t 的唯一路径，按照节点标签执行查询所产生的总成本。

3. 标准二分搜索的近似保证

这里我们证明，标准二分搜索算法始终查询剩余区间的中点，它对于任意对称单调成本函数 h都是一个 4-近似算法，从而证明定理 1。分析中得到的 4 倍因子可能并非紧确。不过，正如我们在第 3.1 节所示，即使当 $h(x) = x$ 是线性的，二分搜索也无法达到优于 $4/(\sqrt{33}-3) \approx 1.457$ -近似的保证；对于一般成本函数，它也无法达到优于 2-近似的保证。

我们用 $\mathcal{B}$ 表示二分搜索算法，并先给出 $\mathcal{B}$ 所产生总成本的一个直接上界：

$$C(\mathcal{B}) \le \sum_{i=1}^{\lfloor \log_2(n) \rfloor} h\left(\lfloor n/2^i \rfloor\right).$$

为证明这个上界，注意长度为 ℓ 的路径的中位数到路径上任意点的距离至多为 $\lfloor \ell/2 \rfloor$。当二分搜索进行第 i^{th} 次查询时，剩余子路径的长度至多为$\lfloor n/2^{i-1} \rfloor$。因此，第 i^{th} 次查询的成本至多为 $h\left(\lfloor \lfloor n/2^{i-1} \rfloor /2 \rfloor\right) \le h\left(\lfloor n/2^i \rfloor\right)$。当剩余子路径的长度为 1（或 0）时，剩余成本为 0。因此，总成本产生于满足 $\lfloor n/2^{i-1} \rfloor \ge 2$ 的轮次 i，这些轮次为 $i \le \log_2 n$。

证明的关键在于，为任意算法（不失一般性地，假设其为规范算法） $\mathcal{A}$建立成本下界。我们将证明以下三个界：

$$C(\mathcal{A}) \ge h(\lfloor n/2 \rfloor) \tag{1}$$

$$C(\mathcal{A}) \ge h(\lfloor n/4 \rfloor) + h(\lfloor n/8 \rfloor) \tag{2}$$

$$2C(\mathcal{A}) \ge \sum_{i=4}^{\lfloor \log_2(n) \rfloor} h\left(\lfloor n/2^i \rfloor\right). \tag{3}$$

利用这些界，近似保证 4 便可直接得出：

$$\begin{aligned} 4C(\mathcal{A}) &\ge h(\lfloor n/2 \rfloor) + (h(\lfloor n/4 \rfloor) + h(\lfloor n/8 \rfloor)) + \sum_{i=4}^{\lfloor \log_2(n) \rfloor} h\left(\lfloor n/2^i \rfloor\right) \\ &= \sum_{i=1}^{\lfloor \log_2(n) \rfloor} h\left(\lfloor n/2^i \rfloor\right) \ge C(\mathcal{B}). \end{aligned}$$

方程（1）给出的界很直接：对于 q_1 的查询，对手选择 $t = 1$ 或 $t = n$，取距离 q_1 更远的那个。该节点与 $\lfloor n/2 \rfloor$ 的距离至少为 q_1，因此仅第一步就会产生 h （$\lfloor n/2 \rfloor$）的成本。

方程 (2) 的下界也相当直接。对于查询 q_1 （不失一般性，设 $q_1 \ge 1 + \lfloor n/2 \rfloor$），对手指向较大的剩余集合所在的方向；此处为左侧。如果下一次查询是 $q_2 \ge \lfloor n/8 \rfloor + 1$，则目标被确定为位于左端点，总成本至少为 $h(q_1 - 1) + h(q_2 - 1) \ge h(\lfloor n/2 \rfloor) + h(\lfloor n/8 \rfloor) \ge h(\lfloor n/4 \rfloor) + h(\lfloor n/8 \rfloor)$。否则（当 $q_2 \le \lfloor n/8 \rfloor$时），目标被确定为 $\lceil n/4 \rceil$。因此，总成本至少为 $h(1 + \lfloor n/2 \rfloor - \lceil n/4 \rceil) + h(\lceil n/4 \rceil - \lfloor n/8 \rfloor) \ge h(\lfloor n/4 \rfloor) + h(\lfloor n/8 \rfloor)$。

方程 (3) 的下界要复杂得多。我们将分析这样一种对手：对于任何查询 q，它总是指向较大的剩余子路径所在的方向。需要指出的是，当成本函数不是常数时，这种对手策略未必是最优的（即最坏情况）；例如，即使这样会导致子路径变短，对手通常也更应该指向远离更多先前查询的一侧。然而，即使这种并非最坏情况的对手策略，也足以证明所声称的下界。

设 t 为最后揭示的目标。设 $j=0$、.....、ℓ 为 $\mathcal{A}$ 的迭代；对于每个 j，设 L_j、R_j 满足：在进行 j 次查询后，可行集为 $[L_j+1, R_j-1]$；也就是说，L_j、R_j 是紧邻可行集左侧和右侧的不可行节点。可行区间持续缩小这一事实意味着

$$0 = L_0 \le L_1 \le \cdots \le L_\ell = t-1 < t+1 = R_\ell \le R_{\ell-1} \le \cdots \le R_1 \le R_0 = n+1.$$

对于每个 j，设 z_j 为 L_j 和 R_j 中距离 t 最远的那个（若二者距离相同，则任选其一），并令 $d_j = d(t, z_j) = |t - z_j|$。

首先，我们声称 $j(0) = 1$ 满足 $\frac{n}{4\cdot 3^0} < d_{j(0)}$。这是因为，对手总是选择指向剩余可行集较大一侧的方向，因此保证了 $R_1 - L_1 > n/2$，所以 L_1、R_1 中至少有一个与 t 的距离大于 $n/4$。

接下来，我们声称，对于每个 $i = 1, \ldots, 1+\lfloor \log_3(n/4) \rfloor$，都存在[6]一个 $j = j(i) \ge 1$ 满足 $\frac{n}{4\cdot 3^i} < d_j \le \frac{n}{4\cdot 3^{i-1}}$。反设不存在这样的 j。令 j 为满足 $d_j > \frac{n}{4\cdot 3^{i-1}}$ 的最大值；这样的 j 必定存在，因为 $j=1$ 仍是一个选项。另外请注意，i 的上界保证在第 j 步，可行集包含多个节点。由对称性，不失一般性地假设 $z_j = R_j$。若 $|R_{j+1} - t| > \frac{n}{4\cdot 3^{i-1}}$ 或 $|L_{j+1} - t| > \frac{n}{4\cdot 3^{i-1}}$，就会与 j 的最大性矛盾。因此，根据 i 的选取（排除了距离的中间范围），可得 $|R_{j+1} - t| \le \frac{n}{4\cdot 3^i}$ 且 $|L_{j+1} - t| \le \frac{n}{4\cdot 3^i}$。这也意味着 $R_{j+1} < R_j$，因而根据更新过程有 $L_{j+1} = L_j$。因此，查询 j 必定发生在 R_{j+1}，且对手指向左侧，这意味着 $R_{j+1} - L_j \ge R_j - R_{j+1}$。根据三角不等式（以 t 为中间点），有 $R_{j+1} - L_j \le \frac{2n}{4\cdot 3^i}$，所以 $R_j - R_{j+1} \le \frac{2n}{4\cdot 3^i}$ 也成立。再次根据三角不等式（以 R_{j+1} 为中间点），有 $R_j - t \le \frac{n}{4\cdot 3^i} + \frac{2n}{4\cdot 3^i} = \frac{n}{4\cdot 3^{i-1}}$，这与 $d_j > \frac{n}{4\cdot 3^{i-1}}$ 矛盾。

总而言之，到目前为止，我们已经证明，对于每个 $i = 0, \ldots, 1+\lfloor \log_3(n/4) \rfloor$，都存在一个点 $y_i = z_{j(i)}$，满足 $j(i) \ge 1$；在执行 $\mathcal{A}$ 的某个时刻，该点紧邻可行区间的左侧或右侧，且与 t 的距离为 $d_{j(i)} \in (\frac{n}{4\cdot 3^i}, \frac{n}{4\cdot 3^{i-1}}]$（但 $i=0$ 除外，因为我们没有证明其上界）。由于相应的距离区间互不相交，所有点 y_i 都各不相同。此外，每当 $y_i \ne 0$ 和 $y_i \ne n+1$ 时，y_i 紧邻可行区域左侧/右侧这一事实，必定是算法执行过程中查询 y_i 的结果。因此，对于每个满足 $y_i \notin \{0, n+1\}$ 的 i，$\mathcal{A}$ 必定因某次查询而产生至少 $h\left(\lceil \frac{n}{4\cdot 3^i} \rceil\right)$ 的成本，且不同的 i 对应的查询各不相同。由于 $\{0, n+1\}$ 中有一个仅在迭代 0 时是端点，因此 y_i 中至多有一个（而非两个）属于 $\{0, n+1\}$，所以我们可以将 $\mathcal{A}$ 的总成本下界估计如下：

$$C(\mathcal{A}) \ge \sum_{i=0, y_i \notin \{0,n\}}^{1+\lfloor \log_3(n/4) \rfloor} h\left(\left\lceil \frac{n}{4\cdot 3^i} \right\rceil\right) \ge \sum_{i=1}^{1+\lfloor \log_3(n/4) \rfloor} h\left(\left\lceil \frac{n}{4\cdot 3^i} \right\rceil\right) \tag{4}$$

根据 h 的单调性。

最后，我们证明

$$\sum_{i=1}^{1+\lfloor \log_3(n/4) \rfloor} h\left(\left\lceil \frac{n}{4\cdot 3^i} \right\rceil\right) \ge \sum_{i=1}^{\lfloor \log_4(n) \rfloor - 1} h\left(\left\lfloor \frac{n}{4^{i+1}} \right\rfloor\right)$$

$$\sum_{i=1}^{1+\lfloor \log_3(n/4) \rfloor} h\left(\left\lceil \frac{n}{4\cdot 3^i} \right\rceil\right) \ge \sum_{i=1}^{\lfloor \log_4(n/2) \rfloor - 1} h\left(\left\lfloor \frac{n}{2\cdot 4^{i+1}} \right\rfloor\right);$$

然后，将这两个不等式相加，并注意到

$$\sum_{i=1}^{\lfloor \log_4(n) \rfloor - 1} h\left(\left\lfloor \frac{n}{4^{i+1}} \right\rfloor\right) + \sum_{i=1}^{\lfloor \log_4(n/2) \rfloor - 1} h\left(\left\lfloor \frac{n}{2\cdot 4^{i+1}} \right\rfloor\right) = \sum_{i=4}^{\lfloor \log_2(n) \rfloor} h\left(\left\lfloor \frac{n}{2^i} \right\rfloor\right),$$

并将左侧与方程 (4) 相结合，即完成了证明。要看出这两个不等式，只需注意左侧的求和项更多；对于任意给定的 i，我们有 $\lceil \frac{n}{4\cdot 3^i} \rceil \ge \frac{n}{4\cdot 3^i} \ge \frac{n}{4^{i+1}} \ge \lfloor \frac{n}{4^{i+1}} \rfloor$，并且

[6] 可能有多个这样的 j，此时 $j(i)$ 是其中任意一个。

第二个求和也同理（其中，h 函数下的各项又小了 2 倍）。根据 h 的单调性，结论即得。

3.1. 近似比下界。我们证明，对于一般的（对称）单调成本函数，二分搜索至多达到 2-近似；即使成本函数是线性的，近似比也不优于 $4/(\sqrt{33}-3) \approx 1.457$。

为证明 2 的下界，考虑成本函数 $h(x)$，当 $x \geq \lfloor n/4 \rfloor$ 时其值为 1，否则为 0。为避免表述中出现取整[7]，，假设 $n = 2^k - 1$，其中 k。若目标位于 1 处，由于前两次查询的位置是 $\lceil n/2 \rceil$、$\lceil n/4 \rceil$，二分搜索的成本为 2。另一方面，考虑以下（最优）策略。首先在 $\lceil \frac{n}{2} \rceil$处查询；不失一般性，假设目标小于 $\lceil \frac{n}{2} \rceil$。接着，在 $\lceil \frac{n}{6} \rceil$处查询。若目标小于$\lceil \frac{n}{6} \rceil$，该次查询无需成本，随后在剩余区间内进行线性搜索也无需成本。若目标大于$\lceil \frac{n}{6} \rceil$，则在 $\lceil \frac{n}{3} \rceil$处查询。在 $\lceil \frac{n}{6} \rceil$、$\lceil \frac{n}{3} \rceil$和 $\lceil \frac{n}{2} \rceil$ 处查询的合并成本至多为 1，因为对手无法选择一个与这三次查询中的多个位置距离均至少为$\lfloor \frac{n}{4} \rfloor$ 的目标。此时，之后的任何查询成本均为 0。因此，该策略的成本为 1，从而证明二分搜索的近似比下界为 2。

对于线性成本函数 $h(x) = x$，我们考虑在连续区间 $[0,1]$ 上的搜索，并采用相同的搜索和成本模型。这么做仅仅是为了避免表述中出现舍入问题——很容易验证，当 $n \to \infty$ 时，下界会收敛到连续模型对应的下界。首先注意，如果目标位于 0，则二分搜索产生的成本为 $\sum_{i=1}^{\infty} 2^{-i} = 1$。现在考虑以下策略（该策略并非最优）：首先在 $1/2$ 处查询；不失一般性，假设目标小于 $1/2$。然后，在 $\gamma = \frac{\sqrt{33}-5}{8}$ 处查询。如果目标小于 γ，则在区间 $[0,\gamma]$ 上执行二分搜索。如果目标大于 γ，则在$[\gamma, \frac{1}{2}]$上递归搜索。令 C 表示该搜索策略的成本。对于第一种情况，前两次查询的成本至多为 $1/2 + \gamma$，后续二分搜索的成本为 γ，总计为 $1/2 + 2\gamma$。对于第二种情况，前两次查询的总成本恰为 $1/2 - \gamma$（无论目标在区间内的实际位置如何），加上 $C \cdot (1/2 - \gamma)$ 的递归搜索成本；递归搜索是在大小为 $1/2 - \gamma$ 的区间上进行的。这里用到了成本函数的线性性。因此，我们推导出

$$C \leq \max\left(\frac{1}{2} + 2\gamma, \left(\frac{1}{2} - \gamma\right) \cdot (C+1)\right).$$

代入 $\gamma = \frac{\sqrt{33}-5}{8}$ 和 $C = \frac{\sqrt{33}-3}{4}$ 后，我们可以看到这两项都恰好等于 C。因此，C 是该策略成本的上界，由此可得，针对线性代价函数的二分搜索，其近似比的下界为 $1/C = \frac{4}{\sqrt{33}-3} \approx 1.457$。

4. 路径的精确算法

我们假设 成本函数 $h(x)$ 的形式为：当 $x < 0$ 时为 $h(x) = h^-(-x)$，当 $x > 0$ 时为 $h(x) = h^+(x)$，其中

$$h^-(x) = \sum_{m=0}^{p} \alpha_m^- x \qquad\qquad h^+(x) = \sum_{m=0}^{p} \alpha_m^+ x^k$$

是次数至多为 p 的多项式。

为推导出一个最优算法，考虑先前查询中结果偏低的查询集合（记为 Q^-）和结果偏高的查询集合（记为 Q^+）。算法进行的每次查询，其响应都会揭示该查询属于这两个集合中的哪一个。记 $L = \max Q^-$ 和 $R = \min Q^+$；因此，此时目标值的可行集为区间$\{L+1, L+2, \ldots, R-1\}$。

对于任意一组 Q 查询和 $j \in \{0, \ldots, p\}$，记作

$$\beta_j^Q = \sum_{q \in Q} q^j.$$

我们可以将 β_j^Q 视为先前查询序列的摘要。虽然查询序列可能有指数多个，但这些摘要压缩了所有相关信息，因此可能出现的摘要数量只有多项式级别。具体做法是将查询序列概括为重新参数化的成本多项式的系数。

[7]该结果对任意 n均成立。

当目标 get 是 $t \geq L$（这一点有保证），那么 Q^- 中查询的总成本可以写成 a s

$$
\begin{aligned}
c\left(Q^-, t\right) &= \sum_{q \in Q^-} h^-(t-q) = \sum_{q \in Q^-} \sum_{m=0}^{p} \alpha_m^-(t-q)^m \\
&= \sum_{q \in Q^-} \sum_{m=0}^{p} \alpha_m^- \cdot \sum_{j=0}^{m} \binom{m}{j} \cdot (-1)^{m-j} t^j q^{m-j} \\
&= \sum_{j=0}^{p} t^j \cdot \sum_{m=j}^{p} \alpha_m^- \cdot \binom{m}{j} \cdot (-1)^{m-j} \cdot \beta_{m-j}^{Q^-}.
\end{aligned}
$$

通过类似的计算，由于目标为 $t \leq R$， Q^+ 中的查询总成本可以写为

$$
c\left(Q^+, t\right) = \sum_{j=0}^{p} t^j \cdot \sum_{m=j}^{p} \alpha_m^+ \cdot \binom{m}{j} \cdot (-1)^j \cdot \beta_{m-j}^{Q^+}.
$$

首先，给定 $\beta_j^{Q^-}$ 和 $\beta_j^{Q^+}$，目标 t的总成本可以直接计算为 $c(Q^-,t)+c(Q^+,t)$。其次，， ，注意到这些 β_j^Q 是线性的，也就是说，如果 $Q \cap Q' = \emptyset$，那么对于所有 j，都有 $\beta_j^{Q \cup Q'} = \beta_j^Q + \beta_j^{Q'}$。这一事实对于构造递推关系非常有用。第三，由于 Q 最多包含 n 个查询，且每个查询都不超过n，因此所有 β_j^Q 始终都是整数，且不超过 n^{p+1}。

综合这些观察结果，可以采用基于动态规划的方法。子问题由剩余候选项区间的左右端点 L、 R以及成本多项式的系数 $(\beta_j^-)_{j=0}^p$、$(\beta_j^+)_{j=0}^p$来刻画。重要的是，由于这些系数完全决定了目标 t 会导致的成本，因此系数 β_j^-、 β_j^+已充分编码了先前查询 Q，无需再显式记录。

考虑输入实例 $\langle L, R, (\beta_j^-)_{j=0}^p, (\beta_j^+)_{j=0}^p \rangle$。最优解会查询某个 $q \in \{L+1, \ldots, R-1\}$。作为回应，对手判定 $t = q, t < q$，还是 $t > q$。（如果 $q = L+1$ 或 $q = R-1$，则相应的结果 $t < q$和 $t > q$ 是不可能的；我们将通过规定空集的成本为 $-\infty$来表示这一事实。）在这三种情况下，结果如下：

- 如果 $t = q$，则根据先前推导的方程可得成本，结果为

$$
c\left(\langle (\beta_j^-)_{j=0}^p, (\beta_j^+)_{j=0}^p \rangle, q\right) = \sum_{j=0}^{p} q^j \cdot \sum_{m=j}^{p} \binom{m}{j} \cdot \left(\alpha_m^- \cdot (-1)^{m-j} \cdot \beta_{m-j}^- + \alpha_m^+ \cdot (-1)^j \cdot \beta_{m-j}^+\right). \tag{5}
$$

- 如果 $t < q$，则已知目标位于区间 $\{L+1, \ldots, q-1\}$内，而 q过高。因此，对于剩余子问题（其中 R 更新为 q），成本函数由相同的系数 β_j^- 刻画，而 β_j^+ 则更新为 $\beta_j^+ + \beta_j^{\{q\}}$。该子问题可被最优地求解。

- 如果 $t > q$，那么同理可知，目标位于区间 $\{q+1, \ldots, R-1\}$内，而 q过小。因此，对于剩余子问题（将 L 更新为 q后），成本函数仍由相同的系数β_j^+ 刻画，而 β_j^- 则更新为 $\beta_j^- + \beta_j^{\{q\}}$。同样，这个子问题也能得到最优解。

因此，最优成本由以下关于 $R > L+1$ 的递推关系刻画（其中，当 $R \leq L$时，$\mathrm{OPT}(L, R, (\beta_j^-)_{j=0}^p, (\beta_j^+)_{j=0}^p) = -\infty$）：

$$
\begin{aligned}
\mathrm{OPT}(L, R, (\beta_j)_{j=0}^p) = \min_{q=L+1,\ldots,R-1} \max \Big(& c\left(\langle (\beta_j^-)_{j=0}^p, (\beta_j^+)_{j=0}^p \rangle, q\right), \mathrm{OPT}(L, q, (\beta_j^-)_{j=0}^p, (\beta_j^+)_{j=0}^p) + (\beta_j^{\{q\}})_{j=0}^p), \\
& \mathrm{OPT}(q, R, (\beta_j^-)_{j=0}^p + (\beta_j^{\{q\}})_{j=0}^p, (\beta_j^+)_{j=0}^p) \Big).
\end{aligned}
$$

这个动态规划可以按照标准方法自底向上求解（从最小区间到最大区间 $\{L, \ldots, R\}$）。由于系数 β_j^-，β_j^+是至多 n j^{th} 个不超过 n的整数的幂之和，因此其上界为 n^{j+1}；对这 2(p + 1)个系数以及两个端点 L, R 进行编码需要一个大小为 $n^2 \cdot \prod_{j=0}^p n^{2(j+1)} = n^{2+(p+1)\cdot(p+2)} = n^{p^2+3p+2}$的表；每个表项可在 $O(n)$时间内计算，从而得到所声称的界。

这给出了所需的多项式时间算法，并证明了定理 2。

5. 位置相关多项式成本

定理 2 仅适用于成本函数 $f(q,t)$ 仅取决于目标与查询之间距离 的情况。因此，我们的结果不适用于重复发布价格和拥塞控制场景中的遗憾成本函数，因为在该场景下， $q > t$ 的遗憾为 t，因此取决于 t，但并非通过其与 q之间的距离来决定。受此启发，我们研究更一般形式的成本函数，即关于 q 和 t的任意有界次数多项式。为此，我们需要假设该多项式的所有系数也都由 n 的某个多项式所界定。更一般形式的成本函数如下：

$$h^-(q,t) = \sum_{i,j\in\{0,\dots,p\}:i+j\le p} \alpha^-_{i,j} q^i t^j \qquad\qquad h^+(q,t) = \sum_{i,j\in\{0,\dots,p\}:i+j\le p} \alpha^+_{i,j} q^i t^j.$$

所有系数 $\alpha_{i,\ j}$ 均为整数，且其绝对值不超过 $O(n^s)$，其中 s为常数。与第 4 节中的分析类似，查询 Q^-， Q^+ 的总成本可以写成

$$\begin{aligned} c\left(Q^-\cup Q^+,t\right) &= \sum_{q\in Q^-} h^-(q,t) + \sum_{q\in Q^+} h^+(q,t) \\ &= \sum_{q\in Q^-} \sum_{i,j\in\{0,\dots,p\}:i+j\le p} \alpha^-_{i,j} q^i t^j + \sum_{q\in Q^+} \sum_{i,j\in\{0,\dots,p\}:i+j\le p} \alpha^+_{i,j} q^i t^j \\ &= \sum_{j=0}^{p} t^j \cdot \left(\sum_{q\in Q^-} \sum_{k=0}^{p-j} \alpha^-_{j,k} q^k + \sum_{q\in Q^+} \sum_{k=0}^{p-j} \alpha^+_{j,k} q^k \right). \end{aligned}$$

记

$$\gamma_j^{Q^-\cup Q^+} := \sum_{q\in Q^-} \sum_{k=0}^{p-j} \alpha^-_{j,k} q^k + \sum_{q\in Q^+} \sum_{k=0}^{p-j} \alpha^+_{j,k} q^k,$$

可见，对于固定的过去查询集 $Q^-\cup Q^+$，给定目标 t 的成本只需知道 $\gamma_j^{Q^-\cup Q^+}$ 即可计算：

$$c\left(Q^-\cup Q^+,t\right) = \sum_{j=0}^{p} \gamma_j^{Q^-\cup Q^+} \cdot t^j. \tag{6}$$

这些 $\gamma_j^{Q^-\cup Q^+}$ 在相同意义下也关于这些集合呈线性，因此 $\langle L,\ R,\ (\gamma_j)_{j=0}^p \rangle$ 足以提供状态编码，从而可以应用相同的动态规划；揭示目标 t 的成本由方程（6）给出，而非方程（5）。

由于所有 $\alpha^-_{k,\ j}$， $\alpha^+_{k,\ j}$ 均有界于 $O(n^s)$， $Q^-\cup Q^+$ 是一个大小至多为 n 的集合，并且 $q\le n$，所有 γ_j的绝对值均有界于 $O(n^{s+p-j+1})$。因此，表的大小为 $n^2\cdot\prod_{j=0}^{p} n^{s+p-j+1} = O(n^{2+(p+1)\cdot(p+2s+2)/2})$，且每个条目的计算时间为 $O(n)$。这就证明了定理 3。

6. A PTAS for Trees

将第 4 节的方法推广到一般树时，我们面临以下障碍：

(1) T的任意子树都可能成为剩余的可行集。若以显而易见的方式实现动态规划，算法就必须考虑指数多个子树（而对于路径，只需考虑 $O(n^2)$）。(2) 构造用于概括先前查询的多项式系数时，依赖于这样一个事实：每个查询都位于剩余可行区间的左侧或右侧。对于更一般的子树，这种简单的划分（及相应的符号）并不可行。

为应对第一个挑战，我们采用 Berendsohn 和 Kozma [3] 提出的 k-cut STT 框架。 k-cut 搜索树是一种 STT，其中每个可行集都是一个子树，且该子树与其补集之间至多由 k条边分隔。因此，这类子树至多有 $O(n^k)$ 个，我们的算法只需枚举这些子树。关键在于分析这一限制所导致的近似保证损失。为应对第二个

个挑战，我们进一步提出一种方法，对先前查询带来的成本进行编码，使得当被查询的顶点集通过合并各自的子树而合并时，可以将不同集合的编码组合起来。

6.1. 树上的 k-割搜索树。

定义 3 (k-割树，[3])。对于 $k \geq 1$，树 $\mathcal{T}$ 上的 STT $T = (V, E)$ 是一个 k-割树，如果$|\partial_T(\phi_{\mathcal{T}}(u))| \leq k$ 对每个 $u \in \mathcal{T}$都成立；也就是说，在该搜索树下可能出现的每个顶点可行集，都至多由 k 条边与已排除的顶点分隔开（或者等价地，它至多有 k 个相邻顶点）。

我们分析仅关注 k-割 STT 所带来的成本增量。我们的主要引理是对 [3] 的一个结果的如下推广：

引理 5. 设 $T = (V, E)$ 是一棵树，$k \geq 3$, $h\colon \mathbb{N} \to \mathbb{R}$ 是一个单调非递减的成本函数，并且 $f(q,t) = h(d(q,t))$ 是目标为 t 时与查询 q 相关联的成本。设 $\mathcal{T}$ 是 T 上的一个 STT。那么，存在一个 k-割 STT$\mathcal{T}'$，它位于 T 上，并且对所有目标顶点 t:

$$C_{\mathcal{T}'}(t) \leq \left(1 + \frac{1}{\lceil k/2 \rceil - 1}\right) \cdot C_{\mathcal{T}}(t).$$

特别是，与搜索树相关的成本（所有目标值中的最大成本）最多增加到原来的 $\left(1 + \frac{1}{\lceil k/2 \rceil - 1}\right)$ 倍。

为了证明该引理，我们采用 Berendsohn 和 Kozma 提出的一种转换过程 [3]（[3] 中的算法 1），该过程可将一般的 STT 转换为 k-切 STT。他们的目标同样是缩小自适应查询策略的搜索空间，代价是解的质量略有损失。不过，他们只关注一个目标：查询总数；这对应于 $h(d) = 1$ 对所有 $d \geq 1$ 均成立的特殊情况。此时，与 STT $\mathcal{T}$ 相关的最大搜索成本恰好是 $\mathcal{T}$ 的高度。在 [3], 中，Berendsohn 和 Kozma 证明，对于每个 $k \geq 3$，每个位于T 上的 STT $\mathcal{T}$都可以转换为一个位于 T 上的 k-切 STT $\mathcal{T}'$，使得

$$\text{height}(\mathcal{T}') \leq \left(1 + \frac{1}{\lceil k/2 \rceil - 1}\right) \cdot \text{height}(\mathcal{T}).$$

他们构造的核心是对 $\mathcal{T}$ 进行迭代变换。每当某个顶点$u \in \mathcal{T}$的可行集 $\phi(u)$ 的边界大小至少为 k（也就是说，恰好处于违反 k-cut STT 性质的边界上）时，算法就会审慎地选择下一个查询，使其能够分隔可行集边界上的叶节点，从而在下一步中，没有任何可行集包含超过 $k/2$ 个叶节点。具体做法是“提升”对应于所需查询的节点，即将其在搜索树中上移。提升可能会增加其他节点的搜索路径长度，但至多增加 1；而在任何一条路径上，这种提升大约每隔 $k/2$ 步才会发生一次。由此便得到他们所需的界。

我们在引理 5 的证明中借鉴这些主要思路，以适用于更一般的非递减成本函数。具体而言，我们将证明，即使查询成本更一般，同样的构造所导致的常数倍损失也完全相同；我们通过将额外查询导致的成本增加，计入先前执行的一部分查询的成本来实现这一点。

6.2. k-割树上的搜索树。我们介绍以下来自 [3 的关键定义和基本结果].

定义 4（凸包，[4] 的定义9）。设 $T = (V, E)$ 是一棵树。顶点集合 $S \subseteq V$ 的凸包记作 $\text{ch}_T(S)$，是 T 中由 S 中顶点对之间所有路径的并集所诱导的子树，也就是包含 S 中所有元素的 T 的最小连通子集。

定义 5（叶质心，源自 [3] 的定义 6）。设 $T = (V, E)$ 是一棵包含 $n \geq 3$ 个节点的树，其中有 ℓ 个叶节点。若 $T \setminus \{v\}$ 的每个连通分量至多包含 $\left\lfloor \frac{\ell}{2} \right\rfloor + 1$ 个叶节点，则称 T 中的非叶节点 v 为叶质心。

叶质心必定存在，并且可以在线性时间内找到：从任意顶点 v 出发，反复沿着通向叶节点过多的连通分量的方向移动即可。

[3]中证明了关于 STT 可行集边界的以下基本事实。

命题 6（[3]中的引理2）。设 $\mathcal{T}$ 是一棵树 $T = (V, E)$ 上以 r 为根的 STT。那么：

(1) $\partial_T(\phi(u)) \subseteq \{\lambda(u') \mid u' \in P_{\mathcal{T}}(u)\}$. (2) 令 $u \in \mathcal{T}, u \neq r$和 p为 $\mathcal{T}$ 中 u 的父节点。那么，$|\partial_T(\phi(u))| \leq |\partial_T(\phi(p))| + 1$。

命题 6 的第一部分指出，可行集的边界只能包含先前查询过的顶点。第二部分指出，查询一个顶点后，可行集的边界大小至多增加 1。

定义凸包的主要原因是：可行集边界的凸包的叶节点必须是先前查询的子集。

命题 7. 设 $T=(V,E)$ 和 $S\subseteq V$，且诱导子图 $T[S]$ 连通。那么，$\mathsf{ch}_T(\partial_T(S))$ 的叶节点集合等于 $\partial_T(S)$。

证明. 注意，$S':=\mathsf{ch}_T(\partial_T(S))$ 必须是 $S\cup\partial_T(S)$ 的子树。S' 也是 $S\cap S'$ 和 $\partial_T(S)$ 的不相交并。根据凸包的定义，只有 $\partial_T(S)$ 中的顶点才能是 S' 的叶节点，这证明了一个包含方向。反过来，由于 $T[S]$ 连通，$\partial_T(S)$ 中没有顶点能与 S 中多于一个顶点相邻（否则会形成一个环），因此 $\partial_T(S)$ 中的每个顶点实际上都是 S' 的叶节点。□

6.2.1. STT 中的旋转与提升。分析 STT 时，一个非常有用的工具是旋转这一概念。这类旋转与标准搜索树中的旋转基本相同，对应于顶点查询顺序的局部变化，其影响可以单独分析并加以界定。

定义 6（分离性、介于关系）。考虑一棵树 $T=(V,E)$。

(1) 当顶点 v 位于从 u 到 w 的唯一路径上时，我们称 v 分隔 u 和 w（或者说，u 与 w，反之亦然）。等价地，若 u 和 w 位于 $T\setminus\{v\}$ 的不同连通分量中，则称 v 分隔 u 和 w。(2) 若顶点 v 位于 $T\setminus\{u,w\}$ 的连通分量中，且该连通分量与 u 和 w 均相邻，则称顶点 $v\in V$ 介于顶点 $u\neq w\in V$ 之间。我们用 $v\in[u,w]$ 表示这种关系，并用 $[u,w]$ 表示介于 u 和 w 之间的所有顶点组成的集合。

定义 7（STT 上的旋转）。固定一棵树 $T=(V,E)$，并设 $\mathcal{T}$ 是 T 上的一棵 STT。令 $u\in\mathcal{T}$ 为 $\mathcal{T}$ 的一个顶点，p 为其父节点。在 $\mathcal{T}$ 中将 u 朝 p 旋转，得到 STT $\mathcal{T}'$，它具有以下性质：

（1） u 是 $\mathcal{T}'$ 中 p 的父节点。（2） u 的每个满足 $\lambda(u')\in[\lambda(u)\ \ \lambda(p)]$ 的子节点 u'，都会在 $\mathcal{T}'$ 中成为 p 的子节点。（3）对于其他所有顶点对，$\mathcal{T}'$ 中的父子关系与 $\mathcal{T}$ 中的相同。

在 T 为一条路径这一特殊情形下，很容易验证，该定义与搜索树中通常所说的旋转一致。我们不只对 STT 执行单次旋转，还会经常考虑一系列旋转，将特定顶点 u 移动到其某个祖先 x 所在的位置。

定义 8（STT 上的提升）。固定一棵树 $T=(V,E)$，并令 $\mathcal{T}$ 为 T 上的一棵 STT。令 $u\in\mathcal{T}$ 为 $\mathcal{T}$ 的一个顶点，且 x 为 u 的一个祖先。令 $u_0=u$，并令 u_{i+1} 为 $\mathcal{T}$ 中 u_i 的父节点，对所有 i，其中 $u_\ell=x$。将 u 提升至 x（位于 $\mathcal{T}$ 中），通过先朝 u_1 旋转 u，再朝 u_2 旋转，依此类推，最后朝 $u_\ell=x$ 旋转来实现。

以下命题描述了提升对可行集的影响。

命题 8。令 T 为一棵树，$\mathcal{T}$ 为 T 上的一棵 STT。令 $u\in\mathcal{T}$ 为一个（非根）顶点，且 x 为 $\mathcal{T}$ 中 u 的一个祖先。令 $\mathcal{T}'$ 为将 u 提升至 x 后得到的 STT。那么，$\phi_{\mathcal{T}'}(u)=\phi_{\mathcal{T}}(x)$。

证明。当 u 被提升至 x 时，以 u 为根的 $\mathcal{T}'$ 子树中的节点集合 $\mathcal{T}$ 以 x 为根的子树中的节点集合相同。由定义 2 即可立即得到该结论。□

我们尤其关注提升叶质心所产生的影响，[3], 中的以下命题对此进行了刻画：提升叶质心实质上会使可行集的边界大小减半：

命题 9（[3] 的引理 5）。设 $u\in\mathcal{T}$ 使得 $|\partial_T(\phi(u))|\geq k$。设 $v\in V$ 是 $\mathsf{ch}(\partial_T(\phi(u)))$ 的一个叶质心，且 $x\in\mathcal{T}$ 是满足 $\lambda(x)=v$ 的顶点。设 $\mathcal{T}'$ 是通过在 $\mathcal{T}$ 中将 x 提升到 u 而得到的 STT。那么，对于 $\mathcal{T}'$ 中 x 的每个子节点 u'，都有 $|\partial_T(\phi_{\mathcal{T}'}(u'))|\leq\lfloor|\partial_T(\phi_{\mathcal{T}}(u))|/2\rfloor+1$。

[8]等价地，如果存在一条从 v 到 u 且不经过 w 的路径，以及一条从 v 到 w 且不经过 u 的路径，那么 v 位于 u 和 w 之间。另一种等价表述是，v 与某个分隔 u 和 w 的顶点位于 $T\setminus\{u,w\}$ 的同一个连通分量中。

6.3. 引理 5的证明。 我们首先分析一次提升如何影响搜索树的成本。

命题 10. 设 T 是一棵树，$\mathcal{T}$ 是 T 上的一棵 STT。设 $u \in \mathcal{T}$ 是一个（非根）顶点，且 x 是 $\mathcal{T}$ 中 u 的一个祖先。设 $u_0 = u$，并且对于所有 i，u_i 在 $\mathcal{T}$ 中的父节点是 u_{i+1}，其中 $u_\ell = x$。令 $\mathcal{T}'$ 为将 u 提升至 x 后得到的树。于是，在 $\mathcal{T}'$ 中搜索目标 t 所对应的新成本满足以下条件：

- 对于所有目标节点 $t \in \bigcup_{j=1}^{\ell} (\phi_{\mathcal{T}'}(u_j) \setminus [\lambda(u) \ldots \lambda(u_j)])$，成本增加 $f(\lambda(u) \ldots t)$：$C_{\mathcal{T}'}(t) = C_{\mathcal{T}}(t) + f(\lambda(u) \ldots t)$。
- 对于所有其他目标节点，成本保持不变或降低：$C_{\mathcal{T}'}(t) \le C_{\mathcal{T}}(t)$。

证明。每个节点 u_1、……、u_ℓ 的根到节点路径 $P_{\mathcal{T}}(u_i)$ 中都新增了 u。每当目标 t 位于 $\phi_{\mathcal{T}'}(u_i)$ 中时，查询 $\lambda(u)$ 的成本就会产生。对于 $[\lambda(u) \ldots \lambda(u_i)]$, 中的节点，这项成本此前已经产生，因为它们当时位于 $\phi_{\mathcal{T}}(u)$ 中；对于其他节点，这会导致所述的成本增量。对于所有其他目标节点 t，到具有 $\lambda(x) = t$ 的节点 x 的路径 $P_{\mathcal{T}}(x)$ 在 $\mathcal{T}'$ 中保持不变，因此查询序列产生的成本相同。□

接下来，我们分析当与某个节点关联的可行集具有较大边界时，提升一个叶质心所产生的影响。

命题 11. 设 $T = (V, E)$ 是一棵树，且 $\mathcal{T}$ 是 T 上的 STT。设 $x \in \mathcal{T}$ 满足：对于某个 $k \ge 3$，有 $|\partial_T(\phi_{\mathcal{T}}(x))| = k$。设 v 是 $\phi_{\mathcal{T}}(x)$ 的一个叶质心，并设 $u \in \mathcal{T}$ 是 $\mathcal{T}$ 中 x 的后代，满足 $\lambda(u) = v$。设 $\mathcal{T}'$ 是将 u 提升至 x 后得到的 T 上的 STT。那么，对于每个目标顶点 $t \in \phi_{\mathcal{T}'}(u) \setminus \{v\}$，$v$ 都会将 t 与 $\mathsf{ch}_T(\partial_T(\phi_{\mathcal{T}'}(u)))$ 的至少 $\lceil k/2 \rceil - 1$ 个叶节点分隔开；等价地，对于 u 在 $\mathcal{T}'$ 中的至少 $\lceil k/2 \rceil - 1$ 个祖先 u'，它会将 t 与 $\lambda(u')$ 分隔开。

证明。令 $S = \mathsf{ch}_T(\partial_T(\phi_{\mathcal{T}'}(u)))$ 表示 $\mathcal{T}'$ 中 $\phi_{\mathcal{T}'}(u)$ 边界的凸包。由于在 $\mathcal{T}$ 中将 u 提升为 x 会得到相同的可行集，即 $\phi_{\mathcal{T}'}(u) = \phi_{\mathcal{T}}(x)$，因此 S 的叶节点数量为 k。根据命题 7，S 的叶节点全部位于 $\partial_T(\phi_{\mathcal{T}'}(u))$ 中；因此，根据命题 6，这些叶节点必须是 $\mathcal{T}'$ 中 u 的祖先的标签。

根据叶质心的定义，$S \setminus \{v\}$ 的每个连通分量至多包含 $T[S]$ 的 $\lfloor k/2 \rfloor + 1$ 个叶节点。因此，S 中的每个目标 t 都被 v 与 $T[S]$ 的至少 $k - (\lfloor k/2 \rfloor + 1) = \lceil k/2 \rceil - 1$ 个叶节点分隔开来。□

命题 11 指出，若可行集 S 与其补集之间恰好由 T 条 k 边分隔，那么查询 S 的叶质心，而不查询 x，则 $\mathcal{T}$ 会将至少 $\lceil \frac{k}{2} \rceil - 1$ 个先前查询过的顶点与可行集中的每个顶点分隔开。这为引理 5证明的核心收费型论证提供了依据：根据命题 10，提升叶质心所导致的成本增量可以取平均，并分摊给至少 $\lceil k/2 \rceil - 1$ 个先前的查询。该论证利用了成本函数的单调性h。

引理 5的证明。 固定 $k \ge 3$。我们描述一个迭代过程，该过程由 Bose 等人在 [4], 中引入，并推广到 k-割搜索树，见 [3], 中的算法 1；该过程可显式构造出 k-割 STT $\mathcal{T}'$，其输入为 STT $\mathcal{T}$。

在每次迭代中，该过程都会提升边界过大的节点 u 的一个叶质心。具体而言，该过程从 $\mathcal{T}' = \mathcal{T}$ 开始。考虑这样一次迭代：至少存在一个节点 $u \in \mathcal{T}'$，满足 $|\partial_T(\phi_{\mathcal{T}'}(u))| \ge k$。设 u 为具有此性质的节点：u 的所有祖先 x 都满足 $|\partial_T(\phi_{\mathcal{T}'}(x))| < k$。由命题 6，$|\partial_T(\phi_{\mathcal{T}'}(u))| = k$。设 $v \in V$ 为 $\mathsf{ch}(\partial_T(\phi_{\mathcal{T}'}(u)))$ 的一个叶质心，x 为 u 的后代，且满足 $\lambda(x) = v$。该过程随后将 x 提升为 u。图 1 以图示方式展示了这一步。

现在我们分析这些提升对搜索 $t \in \mathcal{T}'$ 的成本所产生的影响；在证明的其余部分中，固定一个 t。由命题 10 可知，在最坏情况下，对于 $t \in \phi_{\mathcal{T}}(u)$，将 x 提升为 u 会使搜索 t 的成本增加 $h(d_T(v,t))$。命题 11 保证，$\mathsf{ch}(\partial_T(\phi_{\mathcal{T}}(u)))$ 中至少有 $\lceil k/2 \rceil - 1$ 个叶节点构成的子集与 t 被 v 分隔开。换言之，对于 $\mathcal{T}'$ 中 u 的祖先 u'，至少有 $\lceil k/2 \rceil - 1$ 个节点 $\lambda(u')$ 与 t 被 v 分隔开。我们将此子集记为 S_v，其中 $|S_v| \ge \lceil k/2 \rceil - 1$。

由于 v 将 t 与每个 $v' \in S_v$ 分隔开，因此 v 位于从 t 到 v' 的路径上，这意味着 $d_T(v',t) > d_T(v,t)$。由于 h 单调不减，与提升 x 相关的成本增量，对于每个 $v' \in S_v$，至多为 $h(d_T(v',t))$。由于这个上界对所有 $v' \in S_v$ 都成立，我们可以对 S_v 取上界的平均值，从而得到将 x 提升到 u 的成本增量满足以下界限：

$$h(d_T(v,t)) \le \frac{1}{|S_v|} \cdot \sum_{v' \in S_v} h(d_T(v',t)) \le \frac{1}{\lceil k/2 \rceil - 1} \cdot \sum_{v' \in S_v} h(d_T(v',t)).$$

现在，在 $\mathcal{T}$ 中所有在该过程期间被提升的顶点里，设 x_1、x_2、.....、x_ℓ 是最终位于 $\mathcal{T}'$ 中从根到 t 的路径上的顶点，顺序按它们被提升的先后排列。设 $v_i = \lambda(x_i)$ 为相应的叶质心。我们要指出，关键的是，相应的集合 S_{v_1}、.....、S_{v_ℓ} 两两不相交。

按照该过程中的处理顺序，每当 $i < j$ 时，节点 x_i 都是 x_j 的祖先。由于 $t \in \phi_{\mathcal{T}'}(x_j) \subseteq \phi_{\mathcal{T}'}(x_i)$，所有 $\phi_{\mathcal{T}'}(x_j)$ 都位于 $T \setminus \{v_i\}$ 的同一个连通分量中。根据 S_{v_i} 的定义，目标 t 与 S_{v_i} 被 v_i 分隔；因此，所有 $\phi_{\mathcal{T}'}(x_j)$ 也如此。于是，S_{v_i} 的所有节点与 $\phi_{\mathcal{T}'}(x_j)$ 中的任意节点之间的距离至少为 2；特别地，$S_{v_i} \cap \partial(\phi_{\mathcal{T}'}(x_j)) = \emptyset$。由于 $S_{v_j} \subseteq \partial(\phi_{\mathcal{T}'}(x_j))$，这就意味着所声称的不相交性。

此外，根据命题 6，S_{v_j} 都包含在 $\{\lambda(u) \mid u \in P_{\mathcal{T}}(t)\}$。

将这些事实应用于下文的第二个不等式，我们便可以界定搜索 t 的总成本增量：

$$\begin{aligned} C_{\mathcal{T}'}(t) - C_{\mathcal{T}}(t) &\le \sum_{i=1}^{\ell} \frac{1}{\lceil k/2 \rceil - 1} \cdot \sum_{v' \in S_{v_i}} d_T(v', t) \;=\; \frac{1}{\lceil k/2 \rceil - 1} \cdot \sum_{i=1}^{\ell} \sum_{v' \in S_{v_i}} d_T(v', t) \\ &\le \frac{1}{\lceil k/2 \rceil - 1} \cdot \sum_{v' \in \{\lambda(u) \mid u \in P_{\mathcal{T}}(t)\}} d_T(v', t) \;=\; \frac{1}{\lceil k/2 \rceil - 1} \cdot C_{\mathcal{T}}(t)\,. \end{aligned}$$

因此，对于所有 $t \in V$，$C_{\mathcal{T}'}(t) \le (1 + \frac{1}{\lceil k/2 \rceil - 1}) \cdot C_{\mathcal{T}}(t)$。 ■

6.4. 树上的编码开销成本函数。第二个挑战在于，查询集的“摘要”会变得更加复杂。此前，在评估目标节点 t 的成本时，两组汇总统计量 —— 分别对应左侧和右侧的先前查询 —— 就足够了。重要的是，这些统计量从不需要合并。在树上搜索时，考虑向一个节点发起查询 q 的情况，该节点有多个此前已查询过的子树。为避免搜索空间变得过大，现在应将来自这些不同子树的所有查询汇总为一个“与” q 相关联的统计量。它们各自先前的代表节点与 q 的距离可能各不相同。我们将展示如何通过计算查询集 Q 中各查询到“接口”节点的距离幂之和来生成此类编码；这些接口节点将 Q 与其余可行集分隔开来。

设 $S \subseteq V$ 是某个可行集（T 的一棵子树），并设 $Q \subseteq V$ 为此前查询过的节点集合（因此得到可行集 S）。设 $\bar{v}_1$，.....，$\bar{v}_s \in S$ 为 S 中与 $V \setminus S$ 中某个顶点相邻的顶点。我们将这些顶点称为 S 的接口顶点。对于每个 $i = 1$，.....，s，设 $Q_i \subseteq Q$ 为查询 q 的集合，其中的查询与 S 在 $T \setminus \{\bar{v}_i\}$ 中不属于同一个连通分量，也就是说，这些查询到 S 的路径会经过接口顶点 $\bar{v}_i$。注意，Q_i 构成所有查询 Q 的一个划分。

对于查询集 Q、节点 v（我们主要用此记法表示查询集的匹配对 $Q = Q_i$、$\bar{v} = \bar{v}_i$ 及其对应的接口节点）以及某个 $m \in \{0, \ldots, p\}$，我们定义

$$\beta_m^{Q,v} = \sum_{q \in Q} d_T(q, v)^m.$$

首先请注意，$\beta^{Q,}_{m}{}^{\bar{v}}$是线性的，也就是说，如果 Q, Q' 互不相交，那么 $\beta^{Q \cup Q',}_{m}{}^{v} = \beta^{Q,}_{m}{}^{v} + \beta^{Q',}_{m}{}^{v}$。其次，注意如果 Q 是所有查询的集合，那么对于目标 t，Q 的总成本可以很容易地计算为

$$\begin{aligned} \sum_{q \in Q} f_T(q, t) = \sum_{q \in Q} h\left(d_T(q, t)\right) \;&=\; \sum_{q \in Q} \sum_{m=0}^{p} \alpha_m \cdot d_T(q, t)^m \\ &= \sum_{m=0}^{p} \alpha_m \cdot \sum_{q \in Q} d_T(q, t)^m \;=\; \sum_{m=0}^{p} \alpha_m \cdot \beta_m^{Q,t}. \end{aligned}$$

这也意味着，对于两个集合 Q, Q'，只要 $\beta^{Q,}_{m}{}^{t} = \beta^{Q',}_{m}{}^{t}$对所有 m都成立，那么在这两个查询集下，以 t 为目标的成本相同。从这个意义上说，$\beta^{Q,}_{m}{}^{t}$再次充当摘要；就成本而言，它们涵盖了关于Q的所有相关信息。

现在，考虑一个可行集 S、S 的一个接口顶点 $\bar{v}_i$，以及相应的集合 Q_i。设 $v \in S$ 是一个顶点，它与 $\bar{v}_i$ 的距离为 $r = d(v, \bar{v}_i)$。对于所有 $q \in Q$，从 q 到 v 的路径都必须经过 $\bar{v}_i$，因此 $d(q, v) = d(q, \bar{v}_i) + r$。

因此，我们得到

$$\beta_m^{Q_i,v} = \sum_{q\in Q_i} d_T(q,v)^m = \sum_{q\in Q_i} (d_T(q,\bar{v}_i)+r)^m = \sum_{q\in Q_i}\sum_{j=0}^{m}\binom{m}{j} d_T(q,\bar{v}_i)^{m-j} r^j$$

$$= \sum_{j=0}^{m}\binom{m}{j} r^j \cdot \sum_{q\in Q_i} d_T(q,\bar{v}_i)^{m-j} = \sum_{j=0}^{m}\binom{m}{j}\cdot r^j \cdot \beta_{m-j}^{Q_i,\bar{v}_i}. \tag{7}$$

这意味着，只要对于所有 $q \in Q_i$，$\bar{v}_i$ 都位于从 v 到 q 的路径上，就可以根据顶点 $\bar{v}_i$ 的系数计算顶点 v 的系数 $\beta_m^{Q_i,v}$。因此，对于给定的可行顶点（连通）集合 S，我们将用系数 $(\beta_m^{\bar{v}_i})_{i=1,\dots,s,m=0,\dots,p}$（无需用到这些集合）来编码有关先前查询的可能成本 Q（其成本将在未来揭晓）的所有相关信息。由于我们只会在 k-切 STT 上进行优化，因此有 $s \le k$，所以给定集合 S 的系数总数至多为 $(p+1)k$；而每个系数 $\beta_m^{Q_i,\bar{v}_i}$ 是至多 n 个项的和，每一项都不超过 n^m，因此至多为 n^{m+1}。于是，对于任意给定的可行集 S，系数至多有 $\prod_{j=0}^{p} n^{(j+1)k} = n^{(p+1)(p+2)\cdot k/2}$ 种可能的取值。

6.5. 整合起来：用于 k-割搜索树的动态规划。 作为我们算法的核心部分，现在我们推导递推关系，以计算 T 的子树的最优搜索策略成本。正如引理 5 所证明的，成本最多增加 $1+2/k$ 倍，就可以将 STT 限制为 k-割 STT。这里我们将采用这一限制，这意味着每个正在考虑的可行集（子树） $S \subseteq V$ 与其补集之间至多由 k 条边分隔。

固定这样一个可行顶点集合 S，设 $\bar{v}_1$、……、$\bar{v}_s \in S$（其中 $s \le k$）为 S 中至少有一个邻居位于 S 外部的顶点。考虑由 S 表征的实例，以及编码 $(\beta_m^{\bar{v}_i})_{i=1,\dots,s,m=0,\dots,p}$。该实例的最优解必须进行某个下一次查询 $q \in S$。如果对手揭示 $t = q$，那么这次额外查询的成本为 0，而之前查询的总成本由 $\sum_{i=1}^{s}\sum_{m=0}^{p}\alpha_m \cdot \beta_m^{\bar{v}_i}$ 给出。

否则，令 $S' \neq \emptyset$ 为对手揭示包含 t 的子树，并令 v' 为 S' 中与 q 相邻的唯一邻居。注意，v' 是 S' 的接口顶点。令 S 中位于 S' 之外的接口顶点的索引为 $I = \{i \in \{1,\dots,s\} \mid \bar{v}_i \notin S'\}$[9]。那么，对于 $i \in I$，从 $q \in Q_i$ 到 S' 中任意顶点的所有路径都必须经过 v'。令 $Q' = \{q\} \cup \bigcup_{i\in I} Q_i$ 为所有过去已查询节点的集合，这些节点到 S' 的路径都必须经过 v'。根据线性性和方程 (7)，对于任意 m，

$$\beta_m^{v'} = \beta_m^{Q',v'} = d(q,v')^m + \sum_{i\in I}\beta_m^{Q_i v'} = d(q,v')^m + \sum_{i\in I}\sum_{j=0}^{m}\binom{m}{j}\cdot d(\bar{v}_i,v')^j \cdot \beta_{m-j}^{\bar{v}_i}$$

可以根据所有可用信息轻易计算得出。于是，给定实例（其中 S）的最优解成本等于由 S'、$(\beta_m^{\bar{v}_i})_{i=1,\dots,s,\ i\notin I,\ m=0,\dots,p}$ 和 $(\beta_m^{v'})_{m=0,\dots,p}$ 所刻画的实例的最优解成本。

对手的响应会使成本最大化；因此，最优成本是第一种情况$t = q$与任意可能的响应 v' 之间的最大成本，其中这些响应是 q 的邻居且位于 S内，从而得到一个非空集S'。反过来，算法会尝试所有候选查询 $q \in S$，使得所有可能产生的新集合 S' 的边界大小至多为 k。这可以通过广度优先搜索在线性时间内轻松检查；[3] 中的引理 7 说明了如何在线性时间内枚举所有此类候选查询。

由此得到一个递推关系，可以按可行集 S大小递增的顺序，以自底向上的方式实现。顶点可行集至多有 $O(n^k)$个，对于每个可行集，需要考虑的 β_m^v 参数设置至多有 $n^{(p+1)(p+2)\cdot k/2}$ 个。因此，该表的大小为 $O(n^{k+(p+1)(p+2)\cdot k/2})$。每个表项都通过对至多 n 个下一次查询 q的选择取最小值来计算，而对手的响应则是对至多 $k \le n$ 种可能响应取最大值。[10] 因此，总运行时间为 $O(n^{2+(p^2/2+3p/2+2)\cdot k})$，当 p 和 k为常数时，它关于 n 是多项式的。

为了获得真正最优 STT 成本的$(1+\epsilon)$-近似（对割大小不作限制），我们需要选择 $k \ge \lceil 2/\epsilon \rceil \ge 1 + 2/\epsilon$。因此，可以在$(1+\epsilon)$时间内得到 $O(n^{2+(p^2/2+3p/2+2)\cdot(1+2/\epsilon)})$-近似。这就完成了定理4的证明。

备注12。虽然定理4 将定理2 的结果扩展到了树的情形，但我们所用的技术（将查询的总成本编码为 Q 中接口顶点到各点距离的幂之和）无法直接用于扩展定理3的结果。原因在于，位置相关成本的结果关键依赖于

[9] $I = \emptyset$是可行的。[10] 查找所需的（常数个）参数的计算最多会产生多对数项，并且在常数时间算术运算模型下只需常数时间。

能够将路径的顶点与整数 $\{1,\ldots,n\}$对应起来，而对于树来说，这一点不再成立。将定理2 扩展为定理4时，我们能够绕过这一困难，因为成本函数仅取决于到目标的距离。

7. 结论

我们提出了一种具有距离相关成本的目标搜索模型，该模型刻画了这样的场景：决策者通过尝试不同的参数设置，旨在找到参数的正确取值。反馈会告知设置过高还是过低，但所产生的成本（不可观测）会随着其与真实值之间的距离增加。我们首先证明，当成本函数对称时（过度估计和估计不足相同幅度会产生相同成本），标准二分搜索是一个 4-近似。随后我们证明，当成本函数是有界次数多项式时，可以在多项式时间内找到最优策略。这些思路还可扩展为：当目标是树中的一个节点且提供方向信息时（成本仍随距离增加），可以得到一个多项式时间近似方案。

我们的研究采用对抗模型。一种自然的替代方案是假设目标是从已知分布中抽取的。事实上，已有研究针对直线上的这一问题进行了探讨，见 [23]。由于在已知分布下可以明确计算任意一次查询的期望成本，因此，直接应用 $O(n^3)$ 动态规划，即可在直线上针对任意成本函数得到最优解。将这一思路与 k-割 STT 相结合，可以直接得到针对树上任意单调成本函数的多项式时间近似方案。事实上，由于可以明确计算一次查询的期望成本，证明还可扩展到任意“路径单调”成本函数 $f(q,t)$，即对于任意一条以 q 为起点的路径，函数值沿路径上的节点单调不减。细节显而易见。

我们的工作引出了许多值得未来研究的问题。或许最迫切的问题是，对于哪些函数 f，可以在多项式时间内在线上最优地解决该问题？将范围限制为常数次数的多项式，对于应用我们的技术至关重要。事实上，在附录B中，我们给出了证据，表明相同的思路无法推广到其他成本函数类别。人们或许可以通过近似一般函数（例如用多项式近似它们），再应用动态规划的思路，来获得近似结果。然而，这类思路似乎只能有限地适用，因为实际上不仅函数本身，其导数也需要得到良好的近似。

对于树，一个自然的问题是：对于任意（或至少更一般的）单调成本函数，能否获得任何近似保证？例如，对于任意（或凸）成本函数，能否得到常数因子近似？另一方面，正如我们此前所提到的，在这种设定下，我们不知道树上的优化问题是否为 NP 难，无法精确求解。人们或许会希望，将二分搜索推广到树上——即始终查询一个节点，使得由此产生的所有连通分量大小都不超过 $n/2$ ——能够带来近似保证。但考虑一条长度为 ℓ 的路径，其一端连接着一个非常大的星形图（叶节点数超过 ℓ），便可看出事实并非如此。广义二分搜索会首先查询星形图的中心；但如果成本函数在 $\ell/2$ 附近急剧增长，那么先在路径上执行标准二分搜索会好得多。

进一步扩展该模型的一个非常自然的方向，是考虑树以外的图结构。一般图上的二分搜索由 [17] 提出，并在近期一系列论文中得到研究 [12, 14, 15, 27]。在该模型中，目标是给定无向图（边可能带权）中的一个未知节点，该图有 n 个节点。对于查询 $q_i \neq t$，算法会获知一个邻居，该邻居位于从 q_i 到 t的最短路径上。正如 [16], 所示，图上的二分搜索为在线学习提供了一个自然的框架：算法提出一种结构（例如二元分类器、排序或聚类），并由此获得关于其提议中错误的信息（例如误分类数据点、顺序错误的点对，或被错误地聚在一起或分开的点）。 [17] 证明，即使在这一通用模型中，也能通过至多 $\log_2(n)$ 次查询找到目标。这种向一般图情形的推广，受到多种应用的启发。其中一种应用是推荐系统，尤其是能够将待推荐内容嵌入图结构中的系统 [37]。假设用户偏好内容的信息由图中的一个节点 t 表示。那么，系统可以通过例如向用户提供内容 q来了解用户偏好的内容。通过用户与 q的交互，系统能够了解用户的偏好，并可将其建模为从 q指向 t 的方向。不过，向用户提供 q 可能会产生成本，而该成本会随着 q 与 t之间距离的增加而上升。例如，反复向用户推荐其不喜欢的内容，可能会导致用户完全停止使用该服务。因而，这种情形很适合纳入我们所考虑的二分搜索推广模型。

考虑到这些潜在动机以及 [16], 中给出的应用，提案的成本随着 G 中提案到正确节点的距离增加，是很自然的。这与以往研究考察的目标不同；以往研究采用的是“标准”的二分搜索目标，即最小化最坏情况下的总

查询次数，不过 [17] 也确立了节点查询成本已知时的困难性结果。这自然促使我们将非均匀成本模型的研究拓展到任意无向图的情形。对于一些有意义的成本函数类别，能否获得非平凡的近似保证（或困难性结果）？即便成本函数恒等于 1（也就是说，目标是最小化查询次数），在 ETH 假设下，也无法在多项式时间内最小化查询次数。然而，目前尚无近似困难性结果，也没有非平凡的正面近似结果（超出平凡的 $O(\log n)$）；若能针对更一般的成本函数获得此类结果，将会非常有意义。

标准二分搜索的另一种经过广泛研究的推广情形是：算法收到的答案可能存在错误。事实上，针对这一问题的最优算法已在不同的错误模型下，分别针对直线、树和一般图进行了研究。[2, 14, 15, 18, 24]。将非均匀成本模型与算法收到的错误答案结合起来，是未来工作一个自然的方向。

最后，若能明确刻画各种“自然”成本函数对应的最优搜索成本，将很有意义，尤其是线性函数 $h(x) = x$。如附录A所述，我们的实验表明，成本会收敛到 $\approx 0.6245\cdots\cdot n$；尽管我们已经推导出一组递归方程，其解可给出精确的近似常数，但目前尚未能求得解。

参考文献

[1] Yosi Ben-Asher、Eitan Farchi 和 Ilan Newman。《树中的最优搜索》。《SIAM计算杂志》，28(6):2090–2102，1999。[2] Michael Ben-Or 和 Avinatan Hassidim。《贝叶斯学习器在含噪二分搜索中是最优的（在量子情形下也相当不错）》。载于：第49届 IEEE 计算机科学基础研讨会论文集，第221–230页，2008。[3] Benjamin A. Berendsohn 和 L´aszl´o Kozma。《树上的伸展树》。载于：第33届 ACM-SIAM 离散算法研讨会论文集，第1875–1900页，2022。网址：https://epubs.siam.org/doi/abs/10.1137/1.9781611977073.75，arXiv：https://epubs.siam.org/doi/pdf/10.1137/1.9781611977073.75，doi:10.1137/1.9781611977073.75。[4] Prosenjit Bose、Jean Cardinal、John Iacono、Grigorios Koumoutsos 和 Stefan Langerman。《树上的竞争性在线搜索树》。载于：第31届 ACM-SIAM 离散算法研讨会论文集，第31届 ACM-SIAM 离散算法研讨会论文集，第1878–1891页，美国，2020。工业与应用数学学会。[5]Renato Carmo、Jair Donadelli、Yoshiharu Kohayakawa 和 Eduardo S. Laber。《随机偏序集中的搜索》。《理论计算机科学》，321(1):41–57，2004。拉丁美洲理论信息学。网址：https://www.sciencedirect.com/science/article/pii/S0304397504001203，doi:https://doi.org/10.1016/j.tcs.2003.06.001。[6] Ferdinando Cicalese、Tobias Jacobs、Eduardo Laber 和 Marco Molinaro。《树与偏序结构中搜索的复杂性》。《理论计算机科学》，412(50):6879–6896，2011。[7] Ferdinando Cicalese、Tobias Jacobs、Eduardo S. Laber 和 Caio Valentim。《加权树上的二元识别问题》。载于 Frank Dehne、John Iacono 和 J¨org-R¨udiger Sack 编，《算法与数据结构》，第255–266页，柏林、海德堡，2011。施普林格·柏林·海德堡。[8] Ferdinando Cicalese、Tobias Jacobs、Eduardo S. Laber 和 Caio D. Valentim。《加权树上的二元识别问题》。《理论计算机科学》，459:100–112，2012。[9] Ferdinando Cicalese、Bal´azs Keszegh、Bernard Lidick´y、D¨om¨ot¨or P´alv¨olgyi 和 Tom´a˘s Valla。《非均匀成本下的树搜索问题》。《理论计算机科学》，647，2014年4月。doi:10.1016/j.tcs.2016.07.019。[10] Erik D. Demaine、Dion Harmon、John Iacono 和 Mihai P˘atras

37 1 240 251, cu。2007 10 1137 0097539705447347, 10 1137 0097539705447347。《动态最优性——近乎实现》。《SIAM计算杂志》，( ): – 。arXiv：https://doi.org/ . /S doi: . /S 。[11] Dariusz Dereniowski。《偏序中的边排序与搜索》。《离散应用数学》，156(13):2493–2500，2008。[12] Dariusz Dereniowski、Przemysław Gordinowicz 和 Paweł Prałat。《边查询与点对查询——随机图与复杂性》，2022。网址：https://arxiv.org/abs/2203.06006，doi:10.48550/ARXIV.2203.06006。[13] Dariusz Dereniowski、Adrian Kosowski、Przemysław Uzna´nski 和 Mengchuan Zou。《加权树上广义二分搜索的近似策略》。载于：第44届自动机、语言与编程国际研讨会论文集，2017。[14] Dariusz Dereniowski、Aleksander Łukasiewicz 和 Przemysław Uzna´nski。《通过中位数近似在图中进行高效含噪二分搜索》。载于 Paola Flocchini 和 Lucia Moura 编，《组合算法》，第265–281页，Cham，2021。施普林格国际出版。[15] Dariusz Dereniowski、Stefan Tiegel、Przemysław Uzna´nski 和 Daniel Wolleb-Graf。《存在错误时在图中搜索的框架》，2018。网址：https://arxiv.org/abs/1804.02075，doi:10.48550/ARXIV.1804.02075。[16] Ehsan Emamjomeh-Zadeh 和 David Kempe。《鲁棒交互式学习的通用框架》。载于：第31届神经信息处理系统进展会议论文集，第7082–7091页，2017。[17] Ehsan Emamjomeh-Zadeh、David Kempe 和 Vikrant Singhal。《图中的确定性与概率二分搜索》。载于：第48届 ACM 计算理论研讨会论文集，第519–532页，2016。[18] Uriel Feige、Prabhakar Raghavan、David Peleg 和 Eli Upfal。《利用含噪信息进行计算》。《SIAM计算杂志》，23(5):1001–1018，1994。[19] Phyllis G. Frankl 和 Elaine J. Weyuker。《分支测试的可证明改进》。《IEEE软件工程汇刊》，19(10):962–975，1993。doi:10.1109/32.245738。[20] Brent Heeringa、Marius C˘at˘alin Iordan 和 Louis Theran。《动态树状偏序中的搜索》。载于：第12届算法与数据结构国际会议论文集（WADS’11），第512–523页，柏林、海德堡，2011。施普林格出版社。

[21] Tobias Jacobs、Ferdinando Cicalese、Eduardo S. Laber 和 Marco Molinaro。《树中搜索的复杂性：平均情形最小化》。载于 Samson Abramsky、Cyril Gavoille、Claude Kirchner、Friedhelm Meyer auf der Heide 和 Paul G. Spirakis 编，《自动机、语言与编程》，第527–539页，柏林、海德堡，2010。施普林格·柏林·海德堡。[22] Charles Jordan 和 Harry C. Carver。《有限差分演算》。AMS 切尔西出版丛书。Chelsea Publishing Company，1965。网址：https://books.google.com/books?id=3RfZOsDAyQsC。[23] Richard Karp、Elias Koutsoupias、Christos Papadimitriou 和 Scott Shenker。《拥塞控制中的优化问题》。载于：第41届 IEEE 计算机科学基础研讨会论文集，第66–74页，2000。doi:10.1109/SFCS.2000.892066。[24] Richard M. Karp 和 Robert Kleinberg。《含噪二分搜索及其应用》。载于：第18届 ACM-SIAM 离散算法研讨会论文集，第881–890页，2007。[25] Robert Kleinberg 和 F. Thomson Leighton。《了解需求曲线的价值：在线固定价格拍卖中的遗憾界》。载于：第44届 IEEE 计算机科学基础研讨会论文集，第594–605页，2003。doi:10.1109/SFCS.2003.1238232。[26] William J. Knight。《具有可变探测成本的有序数组搜索》。《SIAM计算杂志》，17(6):1203–1214，1988。arXiv：https://doi.org/10.1137/0217076，doi:10.1137/0217076。[27] Akshay Krishnamurthy、Thodoris Lykouris、Chara Podimata 和 Robert Schapire。《非理性智能体环境下的情境搜索》。载于：第53届 ACM 计算理论研讨会论文集（STOC 2021），第910–918页，纽约，美国，2021。计算机协会。doi:10.1145/3406325.3451120。[28] Tak Wah Lam 和 Fung Ling Yue。《图的边排序是困难的》。《离散应用数学》，85(1):71–86，1998。[29] Tak Wah Lam 和 Fung Ling Yue。《在线性时间内实现树的最优边排序》。《Algorithmica》，30(1):12–33，2001。[30] Nathan Linial 和 Michael E. Saks。《信息界适用于有序数据结构上的搜索问题》。载于：第24届 IEEE 计算机科学基础研讨会论文集，第473–475页，1983。doi:10.1109/SFCS.1983.27。[31] Nathan Linial 和 Michael E. Saks。《每个偏序集都有一个中心元素》。《组合理论杂志 A 辑》，40(2):195–210，1985。网址：https://www.sciencedirect.com/science/article/pii/0097316585900871，doi:https://doi.org/10.1016/0097-3165(85)90087-1。[32] Nathan Linial 和 Michael E. Saks。《有序结构搜索》。《算法杂志》，6(1):86–103，1985。[33] Shay Mozes、Krzysztof Onak 和 Oren Weimann。《在线性时间内找到最优树搜索策略》。载于：第19届 ACM-SIAM 离散算法研讨会论文集，第1096–1105页，2008。[34] Gonzalo Navarro、Ricardo Baeza-Yates、Eduardo Barbosa、Nivio Ziviani 和 Walter Cunto。《具有非均匀成本的二分搜索及其在文本检索中的应用》。《Algorithmica》，27:145–169，2000年6月。doi:10.1007/s004530010010。[35] Krzysztof Onak 和 Pawel Parys。《二分搜索的推广：树与类森林偏序中的搜索》。载于：第47届 IEEE 计算机科学基础研讨会论文集，第379–388页，2006。[36] Daniel Sleator 和 Robert E. Tarjan。《列表更新与分页规则的摊销效率》。《ACM通讯》，28:202–208，1985。[37] Shoujin Wang、Liang Hu、Yan Wang、Xiangnan He、Quan Z. Sheng、Mehmet A. Orgun、Longbing Cao、Francesco Ricci 和 Philip S. Yu。《基于图学习的推荐系统：综述》。《ArXiv》，abs/2105.06339，2021。

附录 A. 特定搜索策略：示例说明

本节通过直线上的例子，展示动态规划发现的非平凡策略类型。我们重点研究两种自然的成本函数：线性成本（$h(x) = x$），以及单件商品定价所产生的遗憾，其形式为。 $f(q,t) = \begin{cases} t & \text{若} t - q \\ & q \le t \text{若。} \end{cases}$

示例展示了标准二分搜索与针对特定成本函数进行调整的复杂策略之间的差异。

图 2 展示了最优搜索策略 $h(x) = x$ 和 $n = 10$；其成本为 6。相比之下，二分搜索的成本为 8。最优策略优于二分搜索，因为它会查询路径上“探索较少”的部分中的顶点；也就是说，另一侧的查询次数越多，就越倾向于将查询偏向路径的一侧，以防止多次查询各自造成高昂成本。根据实验结果，从渐近角度来看[11]，我们观察到，长度为 n 的路径上的最优搜索策略成本趋于 $\approx 0.6245n$，而二分搜索的成本趋于 n。

图 3 展示了以下情况下的最优搜索策略：当 $f(q,t) = t$ 时，若 $t < q$ 且 $f(q,t) = t - q$ 时，若 $q \le t$。该最优策略的成本为 17，而二分搜索的成本为 23。请注意，该策略本质上是在较小区间内进行线性搜索；总体而言，其结构与 [25], 中给出的策略相似：先采用“线性式”搜索，更快地确定正确的“范围”，再采用“线性式”搜索，在该范围内进一步缩小搜索范围。

附录 B. 动态规划方法的局限

这里，我们提出一些有力的启发式论据，说明我们的动态规划方法无法扩展到有界度的多项式之外 h。回想一下，使动态规划方法成为可能的关键思路，是为函数 $h^Q(t)$ 选择一种简洁的编码；该函数表示当目标揭示为 t 时，查询 Q 将追溯产生的成本。

在对 h^Q 进行编码时，我们使用单项式基，并保留先前查询中所有单项式的系数。更一般地，可以考虑通过函数基 $B = \{g_1, g_2, \ldots, g_m\}$ 进行编码，使得成本函数可以写成

$$h^Q(t) = \sum_{q \in Q} h(q - t) = \sum_{i=1}^{m} c_i g_i(t) \text{ for all } t.$$

那么，系数 c_i 就构成了 Q 的编码。在我们的动态规划方法中，我们具体使用了单项式基 $B = \{1, x, x^2, \ldots, x^p\}$；下面这个关于 $h(x) = x^3$ 的例子展示了实现编码所需的性质：

$$h(x-2) = (x-2)^3 = x^3 - 6x^2 + 12x - 8 \in \text{Span}(\{1, x, x^2, \ldots, x^p\}).$$

具体而言，以下几点至关重要：

(1) 函数 h（t 的函数）可以表示为任意 $q \in \{1, \ldots, n\}$ 的 $\mathbb{Z}$-线性组合 B；(2) h 是单调的，因此 g_i 也必须是单调的；(3) 将我们的方法扩展到函数基 B 后，仍能在多项式时间内运行。

我们证明，对于满足这些性质的任意基 $B = \{g_1, \ldots, g_m\}$，以及由 B 的元素线性组合而成的任意函数 h，h 都必须是一个多项式。我们的证明涉及离散微积分中的一些技巧，相关内容请参阅 [22]。

对于 $i \in \mathbb{Z}$，令 E_i 表示作用于函数的移位算子，使得 $(E_i f)(x) = f(x + i)$。固定一个有限函数集 $B = \{g_1, \ldots, g_m\}$，并将由 B 中元素的线性组合张成的函数向量空间（在 $\mathbb{R}$上）记作 Span(B)。注意，性质（1）意味着 Span(B) 必须在移位算子作用下封闭。特别地，它必须在 E_1作用下封闭。有鉴于此，固定一个 $g_j \in B$。由于 Span(B) 在 E_1作用下封闭，离散导数 $\Delta g_j := g_j(x+1) - g_j(x) = E_1 g_j - g_j$ 属于 Span(B)。因此，由于 Δ 是一个线性算子，Span(B) 在 E_1 作用下封闭，进而意味着它在差分算子 Δ作用下封闭。

[11]尽管可以通过求解规模不断增大的实例来得到经验值 0.6245……，但还存在一个递推关系，它涉及一系列界，这些界针对的是与最优搜索策略所采用的查询结构相关的某些有理函数。利用这一界序列，可以更高效地推导出 0.6245……这一数值。

现在，考虑集合 $D := \{g_j, \Delta g_j, \Delta^2 g_j, \dots\}$。由于 Span($D$) 是 Span($B$) 的子空间，且 Span($B$) 的维数至多为 m，因此 Span(D) 的维数至多为 m。于是，集合 $\{g_j, \Delta g_j, \Delta^2 g_j, \dots, \Delta^m g_j\}$ 线性相关，这意味着存在常数 c_0、……、c_m（不全为零），使得 $\sum_{i=0}^{m} c_i \Delta^i g_j = 0$。

我们利用恒等式将差分算子表示为移位算子的形式

$$(\Delta^k g)(x) = \left[(-1)^k \sum_{\ell=1}^{k} \binom{k}{\ell} \cdot (-E_1)^{k-\ell} g\right](x) = \left[\sum_{\ell=1}^{k} \binom{k}{\ell} \cdot (-1)^{\ell} E_{k-\ell} g\right](x),$$

其中用到了 $E_1^j = E_j$。注意，该恒等式将每个 Δ^i 表示为具有常系数的移位算子多项式，并且 $E_1^j = E_j$。将此表达式代入方程 $\sum_{i=0}^{m} c_i \Delta^i g_j = 0$，得到关于 g_j 的常系数齐次线性差分方程，其解可表示为

$$g_j(x) = \sum_{r=1}^{M} p_r(x) \cdot e^{\alpha_r x} \cdot (\cos(\beta_r x) + i \sin(\beta_r x))$$

对于某个 $M > 0$，α_r，$\beta_r \in \mathbb{R}$，以及多项式 p_r（参见 [22]第 557 页）。

由于性质 2 要求 g_j 单调，因此对所有 i，必有 $\beta_i = 0$。现在假设存在某个 $g = g_j$，它对应着一个 $\alpha_r \neq 0$，并记作 $a = e^{\alpha_r}$。我们将证明，即使集合 $\{h(1) \mid h \in$ 张成空间(B) $x \in \{1, \dots, n\}\}$（即在 $x = 1$处计算所有可能的函数）也具有关于 n的指数规模，这意味着可能函数的取值无法通过简洁的（多项式规模）编码表示。我们先考虑较简单的情形 $a > 1$。记 $z = \lceil \log_a 2 \rceil$。那么，对所有 x，$E_z g(x) = (x+z)^p \cdot a^{x+z} \geq x^p \cdot 2 \cdot a^x = 2g(x)$。令 $\ell = \lfloor (n-1)/z \rfloor$，并考虑系数向量$\boldsymbol{c} \in \{0,1\}^{\ell}$。现在考虑对所有此类 $\boldsymbol{c}$的整数线性组合 $h_{\boldsymbol{c}} = \sum_{j=1}^{\ell} c_j \cdot E_{j \cdot z} g$。根据前面的论证，它们都属于 B 的整数张成空间。由于对所有 x，$E_{(j+1)z}(x) \geq 2E_{jz}(x)$，因此只要 $\boldsymbol{c} \neq \boldsymbol{c'}$，相应的取值便满足 $h_{\boldsymbol{c}}(1) \neq h_{\boldsymbol{c'}}(1)$。这样的取值共有 $2^{\ell} = 2^{\Omega(n)}$ 个，因此张成空间的像中有指数多个取值。

论证的本质对 $a < 1$ 也相同，只有细节上略有不同。在这里，我们将确保对所有 x，$E_z g(x) \leq g(x)/2$。求解 $(x+z)^p a^{x+z} \leq x^p a^x / 2$，或者等价地 $\left(\frac{x+z}{x}\right)^p \leq a^{-z}/2$，我们发现，只需选择 z 使其满足 $z^p \leq a^{-z}/2$。由此可知，$z = \Omega(\log n)$ 已足够。论证的其余部分基本相同：对于上述形式的每个整数线性组合 $\boldsymbol{c}$，值 $h_{\boldsymbol{c}}(1)$ 都各不相同。因此，我们得到张成空间的像中有 $2^{\ell} = 2^{\Omega(n/\log n)}$ 个值。

因此，存在多项式大小编码这一假设排除了任何带有指数因子的函数 g_j，这意味着所有 g_j 都必须是多项式。

CALVIN LENG，南加州大学，洛杉矶，加利福尼亚州 90089，美国 电子邮箱：
cleng@usc.edu

DAVID KEMPE，南加州大学，洛杉矶，加利福尼亚州 90089，美国 电子邮箱：
david.m.kempe@gmail.com

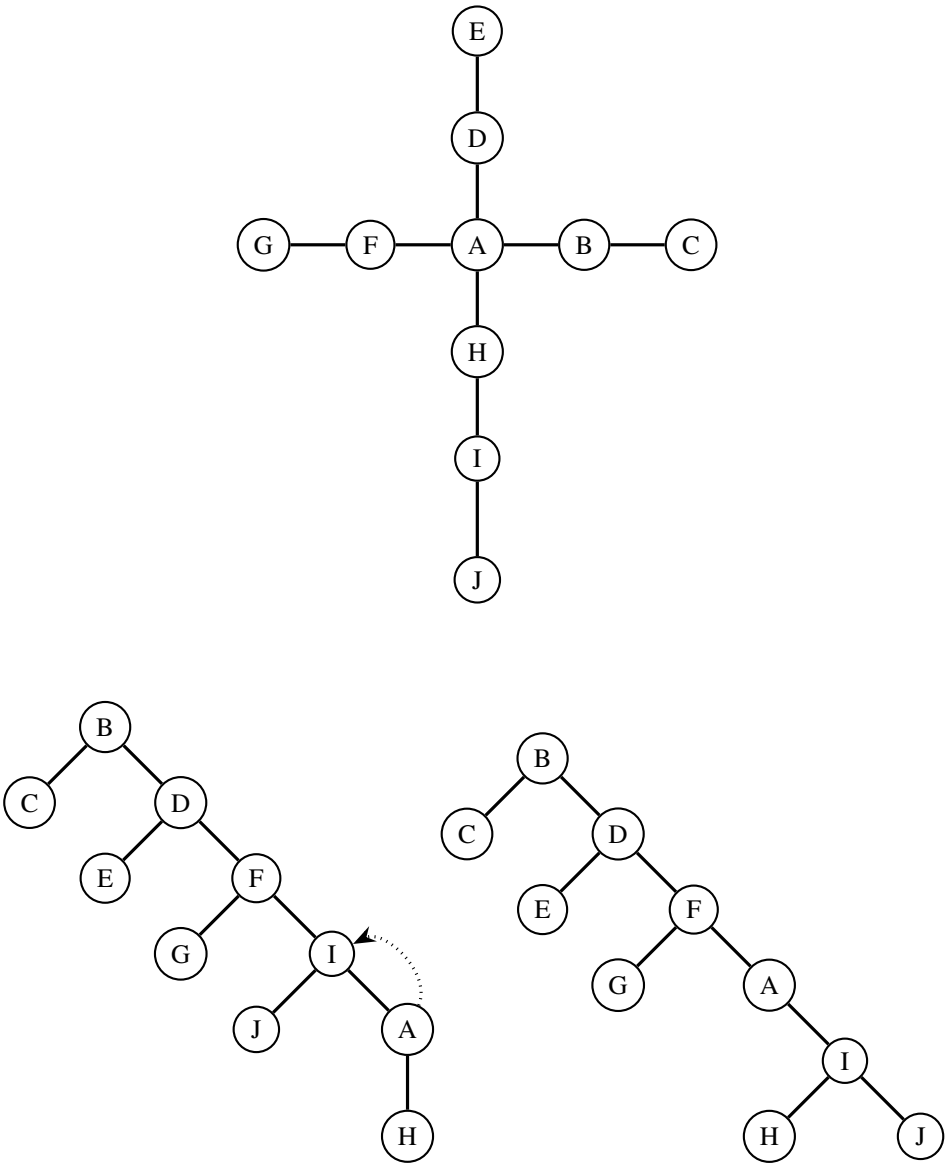


图1. 通过叶质心提升转换为3-割搜索树。（上）：基树 T。（左）：原始搜索树 $\mathcal{T}$，显示了 $\mathcal{T}$ 中每个节点对应的标签（已查询节点）。标签为 I 的顶点，其关联的可行集为 $\phi(I) = \{A, H, I, J\}$，边界大小为3。虽然这暂时不会造成问题，但 I 的子节点（例如 A）的可行集可能会超过边界大小。因此，过程将 $\mathsf{ch}(\partial(\phi(I))) = \mathsf{ch}(\{B, D, F\}) = \{A, B, D, F, H, I, J\}$ 的一个叶质心提升至 I。由于 A 是叶质心（事实上，也是本例中唯一的叶质心），该过程将 A 提升至 I。请注意，叶质心不一定是其所提升到的节点的子节点。此次提升只会使 J 的成本增加 $h(d_T(A, J)) = h(3)$。在定理5的证明中，我们利用这样一个事实：A 将 J 与 B、D 和 F 分隔开来——这些节点已经位于搜索路径 $P_{\mathcal{T}}(J)$ 上 —— 从而将 $h(3)$ 的成本增量均摊到 $\{B, D, F\}$ 中 $\lceil k/2 \rceil - 1 = 1$ 个顶点的成本上。注意，由于分离，所有 $\{B, D, F\}$ 都比 A 离 J 更远。（右）：将 A 提升至 I 后得到的搜索树 $\mathcal{T}'$。此前，$\phi_{\mathcal{T}}(A)$ 的边界大小为4，而提升使 $\phi_{\mathcal{T}'}(A)$ 的边界大小降至1。

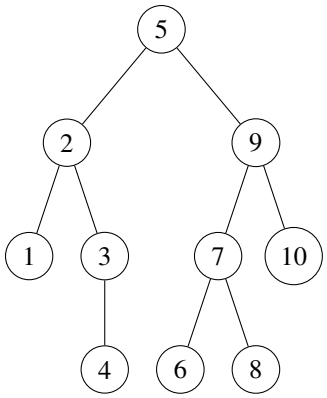


图2. 在具有 $h(x) = x$ 的 10 个顶点路径上的最优搜索策略。

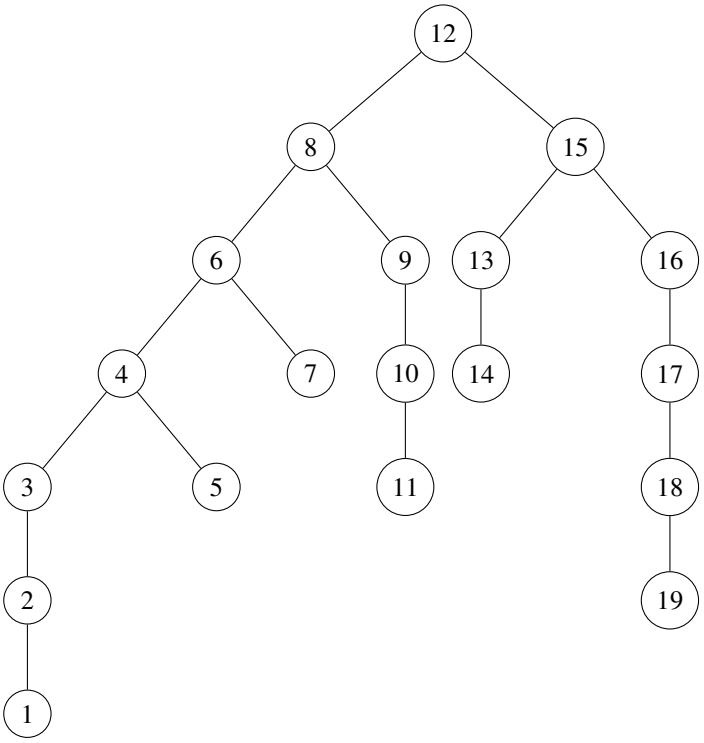


图3. 在具有 $f(q,t) = t \text{ if } t < q \text{ and} f(q,t) = t - q \text{ if } t \geq q$ 的 20 个顶点路径上的最优搜索策略。