Algorithms for the Directed k-Spanner with Minimum Degree Steiner Tree Problem

Hugo Braga

Programa de Pós-Graduação em Mecatrônica Universidadde Federal da Bahia Defesa de Mestrado Orientador: Dr. Flávio Assis

22 de Outubro de 2012

Estrutura da apresentação I

- Motivação
- 2 Por quê o problema é novo?
- Formalização do problema
- 4 Algoritmos
 - APPROX
 - SIM
- Experimentos
- 6 Função Submodular e Matroids
- Conclusão



Problema clássico de árvore Steiner

- O objetivo é encontrar uma árvore que cubra os terminais e que possua custo mínimo.
- Aplicada tipicamente na operação de multicasting.
- Restrições são adicionadas ao problema de multicasting:
 - Requisitos de Qualide de Serviço (QoS).
 - Pode ser modelado através de atraso máximo.
 - Modelado alternativamente através da propriedade de spanner.
 - Limitação no número de vizinhos da comunicação.
 - Restrição de conexões, esforço de replicação de dados, custo de manutenção da topologia em redes móveis.
 - Pode ser generalizado pela propriedade de limitação do grau máximo.

Outras aplicações

- Aplicações de Spanners:
 - Roteamento eficiente.
 - Abondanar o requisito de encaminhar mensagens por caminhos de custo mínimo.
 - Spanners geométricas são utilizadas para diminuir o gasto energético de Redes de Sensores Sem Fio (RSSF).
 - Utilizadas para aproximar distâncias de custo mínimo.
 - Utilizadas por oráculos de distância.
- Restrições de grau em árvores Steiner também existem em cenários onde há a necessidade de balancear a carga na operação de multicasting.

Objetivo

 Buscando modelar o problema de árvore Steiner com as restrições de grau máximo e que possua a propriedade de spanner, nós procuramos tratar no mestrado um novo problema, denominado Árvore Steiner com Grau Mínimo e fator de dilatação k em Grafos Direcionados (Directed k-Spanner with Minimum Degree Steiner Tree Problem - DSMDStP).

Abordagens anteriores dos problemas de Steiner/Spanner/Grau

- Muitos trabalhos assumem espaço euclideano.
- Outros trabalhos consideram grafos não-direcionados.
- Trabalhos que consideram grafos direcionados:
 - Geram árvores geradoras (Spanning trees) ou grafos que não necessariamente são árvores.
 - Tem como objetivo de minização o custo da árvore.
 - Consideram apenas o requisito de minização do grau.

Trabalhos próximos

- [Elkin and Solomon 2009, Elkin and Solomon 2011]:
 - Geram narrow-shallow-low-light trees.
 - Resultado final é uma árvore geradora.
 - Assumem espaço euclideano.
- [Elkin and Kortsarz 2006, Khandekar et al. 2011]:
 - Problema consistem em gerar árvore Steiner que possui grau máximo de saída minimizado em grafos direcionados.
 - O problema em [Khandekar et al. 2011] generaliza o problema abordado em [Elkin and Kortsarz 2006], visto que não impõe a restrição de cobrir todo o conjunto de terminais.
 - Não consideram a propriedade de spanner.
 - O mais próximo que consideram é limitar a altura da árvore.
 - A solução para o problema de Multiple Set-Cover (MSC) definido pelos autores de [Elkin and Kortsarz 2006] é base para a nossa solução.

Definição de DSMDStP

Entrada:

- Um grafo direcionado G = (V, E), onde o peso de cada aresta é definido por uma função C : E → R⁺.
- Um nó de origem s.
- Um fator de dilatação k ($k \in \mathbb{R}^+$, $k \ge 1$).
- Um conjunto de terminais $T \subseteq V \setminus \{s\}$.
- Saída: Encontrar uma arborescência A = (V_A, E_A), com nó origem em s e que cobre T, tal que:
 - $dist(s, t, A) \leq k \cdot dist(s, t, G), \forall t \in T$;
 - o grau máximo de saída dos nós em A, denotado por D_{max}(A) = max_{v∈VA}{odeg(v, A)}, é minimizado.

Impossibilidade de aproximação sublogarítmica: *Background*

- SvMDST é o acrônimo (em inglês) para a versão Steiner do problema de Árvore Geradora com Grau Mínimo em Grafos Direcionados [Fraigniaud 2001].
 - $T \subseteq V$; $s \in T$.
- A prova da impossibilidade é baseada na redução do DSMDStP para o SvMDST.

Teorema 1

[Fraigniaud 2001] A menos que $NP \subset DTIME(n^{\log \log n})$, a solução ótima para o SvMDST não possui aproximação polinomial por $(1-\varepsilon)\log_e|T|$, para qualquer $\varepsilon>0$.





Impossibilidade de aproximação sublogarítmica

Afirmação 1

DSMDStP não admite aproximação polinomial por um fator de $(1 - \varepsilon) \log_e |T|$, para qualquer $\varepsilon > 0$, a menos que $NP \subset DTIME(n^{\log\log n})$.

▶ Prova

- Nosso algoritmo é baseado no algoritmo apresentado em [Elkin and Kortsarz 2006]. Este foi adaptado para considerar a propriedade de spanner.
- Possui duas fases:
 - fase 1: cômputo da \sqrt{I} -partition.
 - fase 2: utilização do problema do Multiple Set-Cover (MSC) para cobrir os demais terminais.
- $V = C \dot{\cup} U$. Seja $UT = U \cap T$ e $CT = C \cap T$.
- Seja I = |T| e d* o grau máximo de uma solução ótima para uma instância de DSMDStP (d* ≤ I).
- Seja G(S) o grafo induzido por um conjunto de nós S.
- Para um nó $u \in U$, Δ -neigh(u) = { $t : t \in UT \land dist(s, u, G) + dist(u, t, G(U)) \le k \cdot dist(s, t, G)$ }.

- Nosso algoritmo é baseado no algoritmo apresentado em [Elkin and Kortsarz 2006]. Este foi adaptado para considerar a propriedade de spanner.
- Possui duas fases:
 - fase 1: cômputo da \sqrt{I} -partition.
 - fase 2: utilização do problema do Multiple Set-Cover (MSC) para cobrir os demais terminais.
- $V = C \dot{\cup} U$. Seja $UT = U \cap T$ e $CT = C \cap T$.
- Seja I = |T| e d* o grau máximo de uma solução ótima para uma instância de DSMDStP (d* ≤ I).
- Seja G(S) o grafo induzido por um conjunto de nós S.
- Para um nó $u \in U$, Δ -neigh(u) = $\{t : t \in UT \land dist(s, u, G) + dist(u, t, G(U)) \le k \cdot dist(s, t, G)\}$

- Nosso algoritmo é baseado no algoritmo apresentado em [Elkin and Kortsarz 2006]. Este foi adaptado para considerar a propriedade de spanner.
- Possui duas fases:
 - fase 1: cômputo da \sqrt{l} -partition.
 - fase 2: utilização do problema do Multiple Set-Cover (MSC) para cobrir os demais terminais.
- $V = C \dot{\cup} U$. Seja $UT = U \cap T$ e $CT = C \cap T$.
- Seja I = |T| e d* o grau máximo de uma solução ótima para uma instância de DSMDStP (d* ≤ I).
- Seja G(S) o grafo induzido por um conjunto de nós S.
- Para um nó $u \in U$, Δ -neigh(u) = { $t : t \in UT \land dist(s, u, G) + dist(u, t, G(U)) \le k \cdot dist(s, t, G)$ }.

- Nosso algoritmo é baseado no algoritmo apresentado em [Elkin and Kortsarz 2006]. Este foi adaptado para considerar a propriedade de spanner.
- Possui duas fases:
 - fase 1: cômputo da \sqrt{I} -partition.
 - fase 2: utilização do problema do Multiple Set-Cover (MSC) para cobrir os demais terminais.
- $V = C \cup U$. Seja $UT = U \cap T$ e $CT = C \cap T$.
- Seja I = |T| e d* o grau máximo de uma solução ótima para uma instância de DSMDStP (d* ≤ I).
- Seja G(S) o grafo induzido por um conjunto de nós S.
- Para um nó $u \in U$, Δ -neigh(u) = { $t : t \in UT \land dist(s, u, G) + dist(u, t, G(U)) < k \cdot dist(s, t, G)$ }.

- Nosso algoritmo é baseado no algoritmo apresentado em [Elkin and Kortsarz 2006]. Este foi adaptado para considerar a propriedade de spanner.
- Possui duas fases:
 - fase 1: cômputo da \sqrt{l} -partition.
 - fase 2: utilização do problema do Multiple Set-Cover (MSC) para cobrir os demais terminais.
- $V = C \cup U$. Seja $UT = U \cap T$ e $CT = C \cap T$.
- Seja I = |T| e d* o grau máximo de uma solução ótima para uma instância de DSMDStP (d* ≤ I).
- Seja G(S) o grafo induzido por um conjunto de nós S
- Para um nó $u \in U$, Δ -neigh(u) = { $t : t \in UT \land dist(s, u, G) + dist(u, t, G(U)) \le k \cdot dist(s, t, G)$ }.

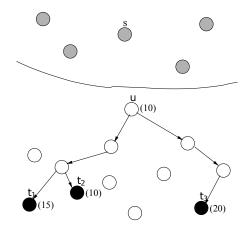
- Nosso algoritmo é baseado no algoritmo apresentado em [Elkin and Kortsarz 2006]. Este foi adaptado para considerar a propriedade de spanner.
- Possui duas fases:
 - fase 1: cômputo da \sqrt{I} -partition.
 - fase 2: utilização do problema do Multiple Set-Cover (MSC) para cobrir os demais terminais.
- $V = C \cup U$. Seja $UT = U \cap T$ e $CT = C \cap T$.
- Seja I = |T| e d* o grau máximo de uma solução ótima para uma instância de DSMDStP (d* < I).
- Seja G(S) o grafo induzido por um conjunto de nós S
- Para um nó $u \in U$, Δ -neigh(u) = { $t : t \in UT \land dist(s, u, G) + dist(u, t, G(U)) \le k \cdot dist(s, t, G)$ }.

- Nosso algoritmo é baseado no algoritmo apresentado em [Elkin and Kortsarz 2006]. Este foi adaptado para considerar a propriedade de spanner.
- Possui duas fases:
 - fase 1: cômputo da \sqrt{I} -partition.
 - fase 2: utilização do problema do Multiple Set-Cover (MSC) para cobrir os demais terminais.
- $V = C \cup U$. Seja $UT = U \cap T$ e $CT = C \cap T$.
- Seja I = |T| e d^* o grau máximo de uma solução ótima para uma instância de DSMDStP ($d^* \le I$).
- Seja G(S) o grafo induzido por um conjunto de nós S.
- Para um nó $u \in U$, Δ -neigh(u) = { $t : t \in UT \land dist(s, u, G) + dist(u, t, G(U)) \le k \cdot dist(s, t, G)$ }.

- Nosso algoritmo é baseado no algoritmo apresentado em [Elkin and Kortsarz 2006]. Este foi adaptado para considerar a propriedade de spanner.
- Possui duas fases:
 - fase 1: cômputo da \sqrt{I} -partition.
 - fase 2: utilização do problema do Multiple Set-Cover (MSC) para cobrir os demais terminais.
- $V = C \cup U$. Seja $UT = U \cap T$ e $CT = C \cap T$.
- Seja I = |T| e d^* o grau máximo de uma solução ótima para uma instância de DSMDStP ($d^* \le I$).
- Seja G(S) o grafo induzido por um conjunto de nós S.
- Para um nó $u \in U$, Δ -neigh(u) = { $t : t \in UT \land dist(s, u, G) + dist(u, t, G(U)) \le k \cdot dist(s, t, G)$ }.

∆-neighbourhood

● *t* = 2



$$d(u, t_1, G(U)) = 15$$

$$d(u, t_2, G(U)) = 12 \Rightarrow d(u, t_3, G(U)) = 25$$

$$d(s, u, G) + d(u, t_1, G(U)) = 10 + 15 \le 2 \cdot 15$$

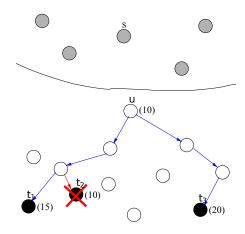
$$d(s, u, G) + d(u, t_1, G(U)) = 10 + 12 > 2 \cdot 10$$

$$\textit{d}(\textit{s},\,\textit{u},\,\textit{G}) + \textit{d}(\textit{u},\,\textit{t}_{1}\,,\,\textit{G}(\textit{U})) = 10 + 25 \, \leq \, 2 \, \cdot \, 20$$



∆-neighbourhood

● *t* = 2



$$d(u, t_1, G(U)) = 15$$

 $d(u, t_2, G(U)) = 12 \Rightarrow$
 $d(u, t_3, G(U)) = 25$

$$d(s, u, G) + d(u, t_1, G(U)) = 10 + 15 \le 2 \cdot 15$$

$$d(s, u, G) + d(u, t_1, G(U)) = 10 + 12 > 2 \cdot 10 \Rightarrow \Delta - neigh(u) = \{t_1, t_3\}$$

$$d(s, u, G) + d(u, t_1, G(U)) = 10 + 25 \le 2 \cdot 20$$





Cômputo da \sqrt{I} -partition (Fase 1)

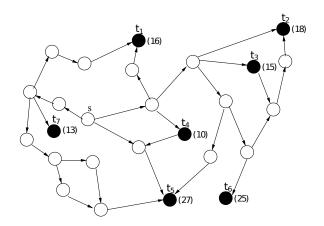
- A \sqrt{I} -partition divide V em dois conjuntos disjuntos: C e U.
- Δ -neigh(u) ($\forall u \in U$) deve conter no máximo \sqrt{I} terminais.
- Partição computada através da eliminação dos \sqrt{I} -bad nodes.
- Um nó u é um \sqrt{I} -bad node se possui mais de \sqrt{I} terminais em Δ -neigh(u).

Procedimento para cômputo da \sqrt{I} -partition (Fase 1)

Algorithm 1: CompPar(G=(V, E), s, k)

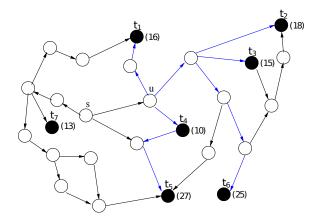
- 1 $C \leftarrow \{s\}, \ U \leftarrow V \setminus \{s\}, \ \textit{Roots} \leftarrow \emptyset$
- **2 while** G(U) contains a \sqrt{I} -bad node V **do**
- Let Cl(v) be the $\lfloor \sqrt{l} \rfloor$ uncovered terminals closest to v in G(U)
- Let T_v be a shortest path arborescence leading from v to CI(v) in G(U)
- $5 \quad | \quad C \leftarrow C \cup V(T_v), \ U \leftarrow U \setminus V(T_v), \ \textit{Roots} \leftarrow \textit{Roots} \cup \{v\}$
- 6 Let $H(V_H, E_H)$ be the forest formed by the union of T_v , ∀v ∈ Roots
- 7 Output (C, U, Roots, H)

CompPar - Grafo de entrada



CompPar - \sqrt{k} -bad node

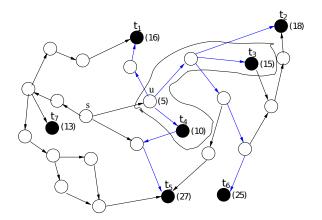
● *t* = 2



- $u \in \sqrt{k}$ -bad node.
- d(s, u, G) = 5; $d(u, t_4, G(u)) = 5$, $d(u, t_5, G(u)) = 10$, $d(u, t_1, G(u)) = 11$, $d(u, t_2, G(u)) = 13$, $d(u, t_6, G(u)) = 20$, $d(u, t_5, G(u)) = 22$.

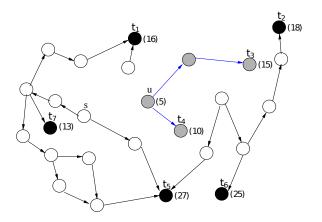
CompPar - $\lfloor \sqrt{I} \rfloor$ terminais mais próximos em UT

● *t* = 2



CompPar - Eliminando os nós cobertos

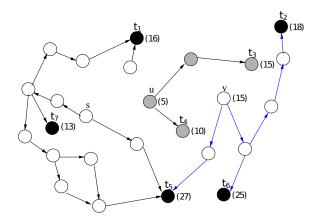
● *t* = 2



• $Root = \{u\}.$



● *t* = 2

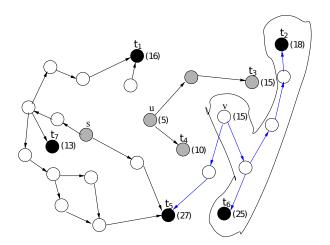


- $v \in \sqrt{k}$ -bad node.
- $d(v, t_6, G(u)) = 10, d(v, t_2, G(u)) = 20, d(v, t_5, G(u)) = 24.$



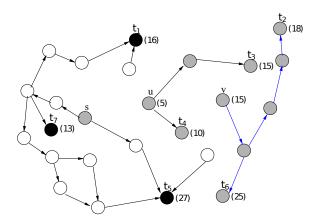
CompPar - $\lfloor \sqrt{I} \rfloor$ terminais mais próximos em UT

● *t* = 2



CompPar - Eliminando os nós cobertos

● *t* = 2



• $Root = \{u, v\}.$



Algoritmo 1: lemas

Lema 1

[Elkin and Kortsarz 2006] O par (C, U) gerado pelo Algoritmo 1 é uma \sqrt{I} -partition.

Lema 2

[Elkin and Kortsarz 2006] |Roots| $\leq \sqrt{I} + 2$.

Construção de caminhos para $t \in CT$

 Algoritmo 2 descreve o procedimento para construção de um grafo que contém caminhos de s para t ∈ CT e que satisfaz k · dist(s, t, G).

Algorithm 2: CompGraphFirstPh(G, s, Roots, H)

- 1 $A_{Roots} \leftarrow SPT(s, Roots, G)$
- 2 $G_{\sqrt{I}-Par} \leftarrow A_{Roots} \cup H(V_H, E_H)$
- $s \ C \leftarrow V(G_{\sqrt{I}-Par}), \ U \leftarrow V \setminus C$
- 4 Output $(C, U, G_{\sqrt{I}-Par})$
 - Terminais em $G_{\sqrt{l-Par}}$ respeitam a restrição de spanner.

Limite no grau máximo de $G_{\sqrt{l-Par}}$ (Algoritmo 2)

Lema 3

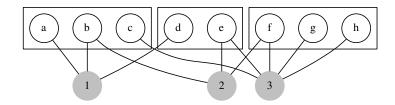
O grau máximo dos nós em $G_{\sqrt{I}-Par}$ é $\leq 2\sqrt{I}+2$

N Dunius

Problema do *Multiple Set-Cover* (MSC)

- Seja $\beta(V_1, V_2, E)$ um grafo bipartite. Um conjunto $S \subseteq V_1$ é denominado um *set-cover* de V_2 se $N(S, \beta) = V_2$.
- Definição formal do MSC:
 - Entrada: Um grafo bipartite $\beta(V_1, V_2, E)$ com $|V_1| + |V_2| = n$. $V_1 = |\dot{I}_{i-1}^d A_i$.
 - Saída: Úm set-cover $S \subset V_1$ de V_2 que minimiza val(S), onde $val(S) = max\{|S \cap A_i|\}_{i=1}^d$.

Exemplo do MSC



- $V_1 = \{a, b, c, d, e, f, g, h\}, V_2 = \{1, 2, 3\}.$
- Os set-covers $S_1 = \{a, e\}$, $S_2 = \{c, d, f\}$ e $S_3 = \{b, f\}$ são soluções ótimas, pois $val(S_1) = val(S_2) = val(S_3) = 1$.

Solução de aproximação para o MSC

Teorema 2

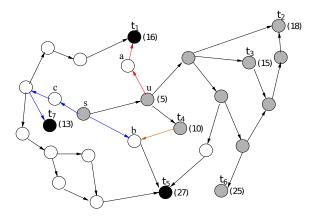
[Chekuri and Kumar 2004]. Existe uma algoritmo guloso com fator de aproximação de $(\log |V_2| + 1)$ para o problema do MSC.

Cobrindo nós em UT através do MSC (Fase 2)

- Utilização de uma instância do MSC para constuir caminhos para os nós em UT ao passo que o grau é limitado (similar a [Elkin and Kortsarz 2006]).
- Fase dividida em duas partes: utilização do MSC para encontrar um subconjunto de nós cobertos; em sequência caminhos para os terminais em UT a partir destes nós são definidos.
- Definição da instância $\beta = (V_1, V_2, \varepsilon)$ do MSC:
 - V_1 é formado por *pseudo nós*. $V_1 = \{x_{u,v} : u \in C, v \in U, (u,v) \in E\}.$
 - $V_2 = UT$.
 - Existe uma aresta em ε entre $x_{u,v}$ e $t \in V_2$ sse $dist(s, u, G) + C(u, v) + dist(v, t, G(U)) \le k \cdot dist(s, t, G)$.
 - Seja $A_u = \{x_{u,v} : v \in N(u, G(U))\}. \ V_1 = \dot{\bigcup}_{u \in C} A_u.$

Instância do MSC

- t = 2
- Caminhos coloridos satisfazem a relação $dist(s, u, G) + C(u, v) + dist(v, t, G(U)) \le k \cdot dist(s, t, G)$.

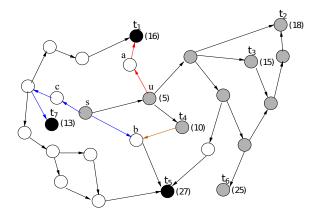


- $V_1 = \{X_{u,a}, X_{t_a,b}, X_{s,b}, X_{s,c}\}$, partição de $V_1 = \{\{X_{u,a}\}, \{X_{t_a,b}\}, \{X_{s,b}, X_{s,c}\}\}$.
- $V_2 = \{t_1, t_5, t_7\}.$
- $\varepsilon = \{(X_{u,a}, t_1), (X_{t_A,b}, t_5), (X_{s,b}, t_5), (X_{s,c}, t_7)\}.$



Instância do MSC

- t = 2
- Caminhos coloridos satisfazem a relação $dist(s, u, G) + C(u, v) + dist(v, t, G(U)) \le k \cdot dist(s, t, G)$.



- $V_1 = \{X_{u,a}, X_{t_d,b}, X_{s,b}, X_{s,c}\}$, partição de $V_1 = \{\{X_{u,a}\}, \{X_{t_d,b}\}, \{X_{s,b}, X_{s,c}\}\}$.
- $V_2 = \{t_1, t_5, t_7\}.$



Limite superior para a solução da instância do MSC

Lema 4

A instância $\beta = (V_1, V_2, \varepsilon)$ do MSC admite uma solução $S^* \subseteq V_1$ t.q. $val(S^*) \leq d^*$.

D

Limite da interseção entre a solução do MSC e cada partição de V_1

Lema 5

Seja D uma solução para $\beta=(V_1,V_2,\varepsilon)$ com $V_1=\dot\bigcup_{v\in C}A_v$ usando o algoritmo apresentado em [Chekuri and Kumar 2004]. Então $\max_{v\in C}|D\cap A_v|=O(\log I)\cdot d^*$.

Algoritmo de Aproximação

Algorithm 3: Algoritmo de Aproximação para o DSMDStP

```
1 Input: G = (V, E), s \in V, T \subset V, k
   Output: A_f = (V_A, E_A)
```

- 2 (C, U, Roots, H) \leftarrow CompPar(G, s, k)
- 3 $(C, U, G_{\sqrt{I}-Par}) \leftarrow CompGraphFirstPh(G, s, Roots, H)$
- 4 Build the MSC instance $\beta = (V_1, V_2, \varepsilon)$
- 5 $D \leftarrow Apply approximation algorithm [Chekuri and Kumar 2004] on$
- 6 $\Gamma(V_{\Gamma}, E_{\Gamma}), V_{\Gamma} \leftarrow \emptyset, E_{\Gamma} \leftarrow \emptyset$
- 7 foreach $t \in V_2$ do

8 Choose
$$x_{u,v} \in D$$
: $(x_{u,v},t) \in \varepsilon$ and $C(u,v) + dist(v,t,G(U))$ is minimum $s \leftarrow sp(v,t,G(U))$

- $V_{\Gamma} \leftarrow V_{\Gamma} \cup \{u\} \cup V(s)$ 10
- $E_{\Gamma} \leftarrow E_{\Gamma} \cup \{(u,v)\} \cup E(s)$
- 12 $G_f \leftarrow G_{\sqrt{I}-Par} \cup \Gamma(V_{\Gamma}, E_{\Gamma})$

 $A \cap A \cap SDT(C T \cap C)$

4 D > 4 P > 4 E > 4 E > 9 Q P

Grau máximo de Gf

Lema 6

O grau máximo dos nós em G_f é $\leq 2\sqrt{l} + 2 + O(\log l) \cdot d^*$.

Custo máximo dos caminhos para os terminais

Lema 7

 $\forall t \in T$, $dist(s, t, G_f) \leq k \cdot (dist(s, t, G) + dist(s, t_{max}, G))$, where $t_{max} \in \{t' | (t' \in T) \land (\forall t'' \in T : dist(s, t'', G) \leq dist(s, t', G))\}$.

▶ Provs

Teorema Principal

Teorema 3

Algoritmo 3 gera uma arborescência A_f com grau máximo $2\sqrt{k} + 2 + O(\log l) \cdot d^*$ e que possui caminhos entre s e cada terminal $t \in T$ com custo $\leq k \cdot (dist(s,t,G) + dist(s,t_{max},G))$.

Complexidade

• $O((\log |T|)(|V|^3|T|^2)$.

▶ Dotalbo

Resumo da heurística

- SIM: acrônimo para Sliced and Iterative Msc.
 - Aplicação iterativa do algoritmo para o MSC.
 - O conjunto de terminais é 'fatiado' e o MSC é aplicado é um conjunto menor se comparado ao algoritmo de aproximação.
- Em cada iteração, os próximos $\lfloor \sqrt{I} \rfloor$ terminais (não cobertos) são cobertos $\rightarrow \textit{Next}_{ter}^{\sqrt{I}}$.

Algorithm 4: SIM - Sliced and Iterative MSC

```
1 Input: G = (V, E), s \in V, T \subset V, k
     Output: A_f = (V_A, E_A)
 2 C \leftarrow \{s\}, U \leftarrow V \setminus \{s\}, Marked \leftarrow \emptyset, I \leftarrow |T|
 3 while UT \neq \emptyset do
         Set V_1, V_2 and \varepsilon using the current values of C. U. Marked and
           Next<sup>√i</sup>
         foreach t: t \in V_2 and \nexists edge (x, t) \in \varepsilon for any x do
              Let x_{u,v} be a pseudo node such that:
                     u \in Marked.
                     u has the smallest out-degree and
                     C(u, v) + dist(v, t, G(U)) \le k \cdot dist(s, t, G)
               V_1 \leftarrow V_1 \cup \{x_{u,v}\}, \varepsilon \leftarrow \varepsilon \cup \{(x_{u,v},t)\}
 7
         D ← Apply approximation algorithm [Chekuri and Kumar 2004]
                       \beta(V_1, V_2, \varepsilon)
         \Gamma(V_{\Gamma}, E_{\Gamma}) \leftarrow \emptyset
 9
         foreach t \in V_2 do
10
              Choose a node x_{u,v} \in D: (x_{u,v}, t) \in \varepsilon and
11
                      C(u, v) + dist(v, t, G(U)) is minimum
               Marked \leftarrow Marked \cup \{u\} \{for which x_{(u,v)}\}
12
                      was chosen in the last line}
              s \leftarrow sp(v, t, G(U))
13
               V_{\Gamma} \leftarrow V_{\Gamma} \cup \{u\} \cup V(s)
14
           E_{\Gamma} \leftarrow E_{\Gamma} \cup \{(u,v)\} \cup E(s)
15
         C \leftarrow C \sqcup V_{\Gamma} \sqcup U \leftarrow \sqcup \sqcup V_{\Gamma}
16
         A_f \leftarrow A_f \cup \Gamma(V_{\Gamma}, E_{\Gamma})
17
```

Existência de um caminho para todo terminal em A_f

Lema 8

Seja A_f a arborescência gerada por SIM. $\forall t \in T$, existe um caminho entre s e t em A_f .

Número de iterações do SIM

Lema 9

O número de iterações do SIM é limitado por $\lfloor \sqrt{I} \rfloor + 2$.

Grafo gerado por SIM é uma arborescência

Lema 10

Seja A_f o grafo gerado por SIM. A_f é uma arborescência.

Custo máximo dos caminhos gerados por SIM

Teorema 4

Seja A_f a arborescência gerada por SIM. $\forall t \in T, dist(s, t, A_f) \leq (\lfloor \sqrt{I} \rfloor + 2) \cdot k \cdot dist(s, t, G).$

Complexidade

•
$$O((\log \sqrt{|T|})(|V|^3|T|\sqrt{|T|})).$$

N Detelle

Parâmetros e detalhes da implementação

- Algoritmos implementados em Java.
- Nós espalhados em uma área de 500x500 no espaço Euclideano.
- Alcance de cada nó é de 125 unidades de medida.
- Faixas de valores:
 - Rede: 60 .. 300.
 - Terminais: 10 .. 50.
 - Fator de dilatação: 1 .. 2.
- 30 simulações para cada cenário.

• Algoritmos:

- DSMDStP é um problema novo.
- Comparamos com o SPT.
 - Satisfaz as restrições do problema.
 - Algoritmo simples e eficiente*.
 - Similaridades com os algoritmos propostos (principalmente APPROX).

- Grau máximo
- Razão de violação do custo (CVR): $\frac{\sum_{Vt \in T_{tot}} \sum_{i \in T_{tot}}^{tot} V_{tot}(S,t,G)}{|T_{tot}|}$
 - Pior caso da razão de violação (MAX CVR)
- Percentagem de terminais que violam restrições (PVT):
 1761 100%.
- Percentagem de cenários violados (PVR)

Algoritmos:

- DSMDStP é um problema novo.
- Comparamos com o SPT.
 - Satisfaz as restrições do problema.
 - Algoritmo simples e eficiente*.
 - Similaridades com os algoritmos propostos (principalmente APPROX).

- Grau máximo.
- Razão de violação do custo (CVR): ^{∑∀t∈T_{vio} k-dist(s,t,G)}
 - Pior caso da razão de violação (MAX CVR)
- Percentagem de terminais que violam restrições (PVT)
 100%
- Percentagem de cenários violados (PVR)



• Algoritmos:

- DSMDStP é um problema novo.
- Comparamos com o SPT.
 - Satisfaz as restrições do problema.
 - Algoritmo simples e eficiente*.
 - Similaridades com os algoritmos propostos (principalmente APPROX).

- Grau máximo.
- Razão de violação do custo (CVR): $\frac{\sum_{\forall t \in T_{vio}} \frac{\textit{dist}(s,t,A_f)}{\textit{k-dist}(s,t,G)}}{|T_{vio}|}$
 - Pior caso da razão de violação (MAX CVR).
- Percentagem de terminais que violam restrições (PVT): ^{|Tyio|}/_T · 100%.
- Percentagem de cenários violados (PVR).



• Algoritmos:

- DSMDStP é um problema novo.
- Comparamos com o SPT.
 - Satisfaz as restrições do problema.
 - Algoritmo simples e eficiente*.
 - Similaridades com os algoritmos propostos (principalmente APPROX).

- Grau máximo.
- Razão de violação do custo (CVR): $\frac{\sum_{\forall t \in T_{vio}} \frac{dist(s,t,A_f)}{k \cdot dist(s,t,G)}}{|T_{vio}|}$
 - Pior caso da razão de violação (MAX_CVR).
- Percentagem de terminais que violam restrições (PVT): ^{|Tyjo|}/_{|TT} · 100%.
- Percentagem de cenários violados (PVR).



• Algoritmos:

- DSMDStP é um problema novo.
- Comparamos com o SPT.
 - Satisfaz as restrições do problema.
 - Algoritmo simples e eficiente*.
 - Similaridades com os algoritmos propostos (principalmente APPROX).

- Grau máximo.
- Razão de violação do custo (CVR): $\frac{\sum_{\forall t \in T_{vio}} \frac{\textit{dist}(s,t,A_f)}{k \cdot \textit{dist}(s,t,G)}}{|T_{vio}|}$
 - Pior caso da razão de violação (MAX CVR).
- Percentagem de terminais que violam restrições (PVT):

 | Tripe | 100%.
- Percentagem de cenários violados (PVR).



Algoritmos:

- DSMDStP é um problema novo.
- Comparamos com o SPT.
 - Satisfaz as restrições do problema.
 - Algoritmo simples e eficiente*.
 - Similaridades com os algoritmos propostos (principalmente APPROX).

- Grau máximo.
- Razão de violação do custo (CVR): $\frac{\sum_{\forall t \in T_{\textit{vio}}} \frac{\textit{dist}(s,t,A_f)}{|T_{\textit{vio}}|}}{|T_{\textit{vio}}|}$
 - Pior caso da razão de violação (MAX_CVR).
- Percentagem de terminais que violam restrições (PVT): $\frac{|T_{vio}|}{|T|} \cdot 100\%$.
- Percentagem de cenários violados (PVR).



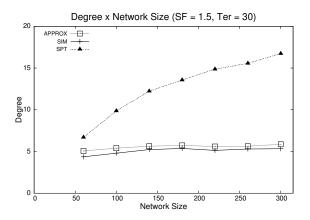
Algoritmos:

- DSMDStP é um problema novo.
- Comparamos com o SPT.
 - Satisfaz as restrições do problema.
 - Algoritmo simples e eficiente*.
 - Similaridades com os algoritmos propostos (principalmente APPROX).

- Grau máximo.
- Razão de violação do custo (CVR): $\frac{\sum_{\forall t \in T_{vio}} \frac{\textit{dist}(s,t,A_f)}{k \cdot \textit{dist}(s,t,G)}}{|T_{vio}|}$
 - Pior caso da razão de violação (MAX_CVR).
- Percentagem de terminais que violam restrições (PVT): ^{|Tyio|}/_{|T|} · 100%.
- Percentagem de cenários violados (PVR).



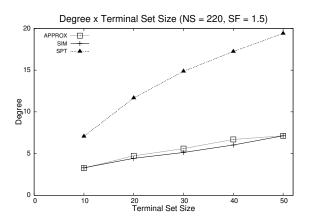
Grau x Rede



- APPROX e SIM escalam bem.
- SPT possui grau 3x maior do que os outros algoritmos em redes densas.



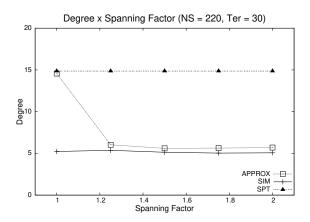
Grau x Terminais



- Crescrimento do grau para APPROX e SIM já era esperado.
 - Em cada fase, o grau máximo do(s) algoritmo(s) é afetado pela quantidade de terminais.
- Taxa de crescimento bem maior para SPT.



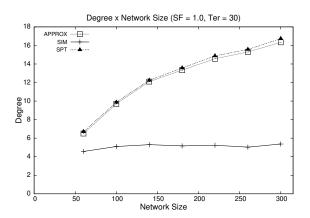
Grau x Fator de Dilatação



- Comportamento de SIM é estável.
- APPROX escala bem, exceto para o caso restritivo do fator de dilatação.

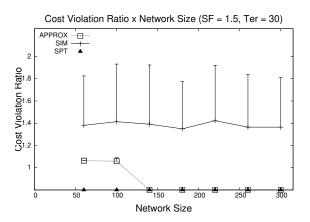


Grau x Rede, SF = 1



- Curva do APPROX assemelha-se à curva do SPT.
 - < sf ⇔ menos opções de caminhos para os terminais não-cobertos.
 - Basicamente os caminhos que sobram fazem parte da árvore de custo mínimo (SPT).

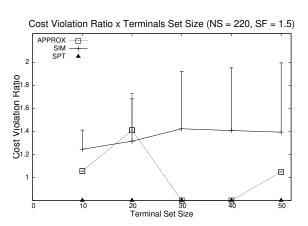
CVR x Rede



- Valores de APROX próximos de 1, ou não há violação.
- SIM apresenta valores uniformes e baixos.
- MAX_CVR abaixo de 2 para o SIM e próximo do CVR para APPROX.

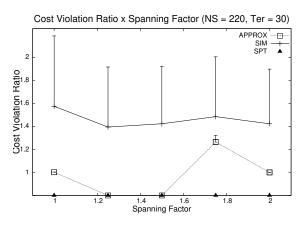


CVR x Terminais



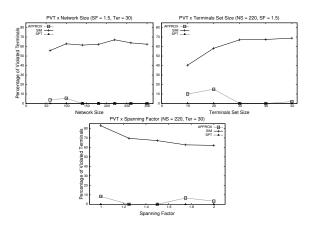
• Comportamente semelhante ao gráfico anterior.

CVR x Fator de dilatação



Comportamente semelhante ao gráfico anterior.

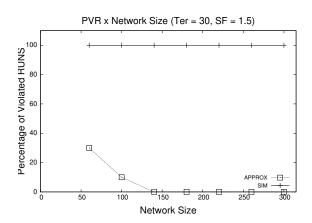
PVT



 Percentagem pequena dos terminais que violam para o APPROX (10%).



PVR x Rede



- APPROX causa violação em uma percentagem relativamente baixa de cenários.
 - Apenas um terminal é suficiente para considerar que ocorreu violação no cenário.



Experimentos - Resultado

- Grau baixo para os algoritmos propostos.
 - Exceto para APPROX no pior cenário com relação ao FD.
 - Neste cenário, APPROX equivale ao SPT com relação ao grau.
- SIM possui melhor grau e escala bem.
- APPROX é melhor que SIM nas métricas que concerne a propriedade de spanner.
- Com APPROX, em metade das situações não ocorreu violação; na outra metade, a violação foi por um fator (CVR) baixo.
- Com relação ao spanner, SIM apresenta bons resultados para a principal métrica (CVR), a métrica qualitativa.
 - a CVR+1.4
 - MAX CVR: 2.

Experimentos - Resultado

- Grau baixo para os algoritmos propostos.
 - Exceto para APPROX no pior cenário com relação ao FD.
 - Neste cenário, APPROX equivale ao SPT com relação ao grau.
- SIM possui melhor grau e escala bem.
- APPROX é melhor que SIM nas métricas que concerne a propriedade de spanner.
- Com APPROX, em metade das situações não ocorreu violação; na outra metade, a violação foi por um fator (CVR) baixo.
- Com relação ao spanner, SIM apresenta bons resultados para a principal métrica (CVR), a métrica qualitativa.
 - CVR: 1.4.

Experimentos - Resultado

- Grau baixo para os algoritmos propostos.
 - Exceto para APPROX no pior cenário com relação ao FD.
 - Neste cenário, APPROX equivale ao SPT com relação ao grau.
- SIM possui melhor grau e escala bem.
- APPROX é melhor que SIM nas métricas que concerne a propriedade de spanner.
- Com APPROX, em metade das situações não ocorreu violação; na outra metade, a violação foi por um fator (CVR) baixo.
- Com relação ao spanner, SIM apresenta bons resultados para a principal métrica (CVR), a métrica qualitativa.
 - CVR: 1.4.

Experimentos - Resultado

- Grau baixo para os algoritmos propostos.
 - Exceto para APPROX no pior cenário com relação ao FD.
 - Neste cenário, APPROX equivale ao SPT com relação ao grau.
- SIM possui melhor grau e escala bem.
- APPROX é melhor que SIM nas métricas que concerne a propriedade de spanner.
- Com APPROX, em metade das situações não ocorreu violação; na outra metade, a violação foi por um fator (CVR) baixo.
- Com relação ao spanner, SIM apresenta bons resultados para a principal métrica (CVR), a métrica qualitativa.
 - CVR: 1.4.MAX CVR: 2

Experimentos - Resultado

- Grau baixo para os algoritmos propostos.
 - Exceto para APPROX no pior cenário com relação ao FD.
 - Neste cenário, APPROX equivale ao SPT com relação ao grau.
- SIM possui melhor grau e escala bem.
- APPROX é melhor que SIM nas métricas que concerne a propriedade de spanner.
- Com APPROX, em metade das situações não ocorreu violação; na outra metade, a violação foi por um fator (CVR) baixo.
- Com relação ao spanner, SIM apresenta bons resultados para a principal métrica (CVR), a métrica qualitativa.
 - GVR: 1.4.MAX CVR: 2

Experimentos - Resultado

- Grau baixo para os algoritmos propostos.
 - Exceto para APPROX no pior cenário com relação ao FD.
 - Neste cenário, APPROX equivale ao SPT com relação ao grau.
- SIM possui melhor grau e escala bem.
- APPROX é melhor que SIM nas métricas que concerne a propriedade de spanner.
- Com APPROX, em metade das situações não ocorreu violação; na outra metade, a violação foi por um fator (CVR) baixo.
- Com relação ao spanner, SIM apresenta bons resultados para a principal métrica (CVR), a métrica qualitativa.
 - CVR: 1.4.
 - MAX CVR: 2.

Funções submodulares e Matroids

 MCG pode ser modelado através dos conceitos de funções submodulares e matroids ([Călinescu et al. 2011]).

Formalismo

- O problema abordado em [Călinescu et al. 2011] foi o da maximização de função submodular sob restrição de uma matroid (SUB-M).
- Este problema já tinha sido abordado em outros trabalhos [Nemhauser et al. 1978, Fisher et al. 1978].
- Os autores em [Călinescu et al. 2011] conseguem generalizar o resultado para qualquer tipo de matroid (incluindo matroid de partição), mantendo a garantia de aproximação ((e/(e - 1))).
 - O resultado dos autores é probabilístico

- O problema abordado em [Călinescu et al. 2011] foi o da maximização de função submodular sob restrição de uma matroid (SUB-M).
- Este problema já tinha sido abordado em outros trabalhos [Nemhauser et al. 1978, Fisher et al. 1978].
- Os autores em [Călinescu et al. 2011] conseguem generalizar o resultado para qualquer tipo de matroid (incluindo matroid de partição), mantendo a garantia de aproximação ((e/(e-1))).
 - O resultado dos autores é probabilístico

- O problema abordado em [Călinescu et al. 2011] foi o da maximização de função submodular sob restrição de uma matroid (SUB-M).
- Este problema já tinha sido abordado em outros trabalhos [Nemhauser et al. 1978, Fisher et al. 1978].
- Os autores em [Călinescu et al. 2011] conseguem generalizar o resultado para qualquer tipo de matroid (incluindo matroid de partição), mantendo a garantia de aproximação ((e/(e - 1))).
 - O resultado dos autores é probabilístico:
 - Garantia do grau passaria a ser probabilística.
 - Para o MSC, o limite no número de iterações para cobrir todos os terminais seria probabilístico.

- O problema abordado em [Călinescu et al. 2011] foi o da maximização de função submodular sob restrição de uma matroid (SUB-M).
- Este problema já tinha sido abordado em outros trabalhos [Nemhauser et al. 1978, Fisher et al. 1978].
- Os autores em [Călinescu et al. 2011] conseguem generalizar o resultado para qualquer tipo de matroid (incluindo matroid de partição), mantendo a garantia de aproximação ((e/(e - 1))).
 - O resultado dos autores é probabilístico:
 - Garantia do grau passaria a ser probabilística.
 - Para o MSC, o limite no número de iterações para cobrir todos os terminais seria probabilístico.



- O problema abordado em [Călinescu et al. 2011] foi o da maximização de função submodular sob restrição de uma matroid (SUB-M).
- Este problema já tinha sido abordado em outros trabalhos [Nemhauser et al. 1978, Fisher et al. 1978].
- Os autores em [Călinescu et al. 2011] conseguem generalizar o resultado para qualquer tipo de matroid (incluindo matroid de partição), mantendo a garantia de aproximação ((e/(e - 1))).
 - O resultado dos autores é probabilístico:
 - Garantia do grau passaria a ser probabilística.
 - Para o MSC, o limite no número de iterações para cobrir todos os terminais seria probabilístico.

- O problema abordado em [Călinescu et al. 2011] foi o da maximização de função submodular sob restrição de uma matroid (SUB-M).
- Este problema já tinha sido abordado em outros trabalhos [Nemhauser et al. 1978, Fisher et al. 1978].
- Os autores em [Călinescu et al. 2011] conseguem generalizar o resultado para qualquer tipo de matroid (incluindo matroid de partição), mantendo a garantia de aproximação ((e/(e - 1))).
 - O resultado dos autores é probabilístico:
 - Garantia do grau passaria a ser probabilística.
 - Para o MSC, o limite no número de iterações para cobrir todos os terminais seria probabilístico.

Conclusão I

- Propomos um novo problema, denominado Árvore Steiner com Grau Mínimo e fator de dilatação k em Grafos Direcionados (DSMDStP).
- Provamos que este problema n\u00e3o \u00e9 aproxim\u00e1vel sublogaritimicamente.
- Propomos duas soluções: um algoritmo de aproximação e uma heurística.
- Provamos as propriedades e garantias de ambos os algoritmos.
- Algoritmo de aproximação:
 - Garantia do grau: $2\sqrt{|T|} + 2 + O(\log |T|) \cdot d^*$.
 - Custo máximo: $\leq k \cdot (dist(s, t, G) + dist(s, t_{max}, G))$.
- SIM:
 - Custo máximo: $(|T| + 2) \cdot k \cdot dist(s, t, G)$.
- Extensa simulação foi apresentada.
 - Ambos os algoritmos apresentam grau muito baixo.



Conclusão II

- SIM possui grau uniforme e apresentou melhores resultados que APPROX e SPT.
- A qualidade da violação apresentada é baixa (1.4 na média).
- APPROX apresentou resultados muito bons para a propriedade spanner:
 - Em metade das situações não ocorreu violação.
 - Na outra metade, a violação muito baixa qualitativamente (em média 1 - praticamente sem violação) e quantitativamente.

- Tentar garantir as restrições de spanner.
- Solução distribuída.
- Modelar a solução através de função submodular e matroids.

- Tentar garantir as restrições de spanner.
- Solução distribuída
- Modelar a solução através de função submodular e matroids.

- Tentar garantir as restrições de spanner.
- Solução distribuída.
- Modelar a solução através de função submodular e matroids.

- Tentar garantir as restrições de spanner.
- Solução distribuída.
- Modelar a solução através de função submodular e matroids.

Definições preliminares

- Algoritmo de aproximação por um fator α [Williamson and Shmoys 2011]: consistem em um algoritmo polinomial tal que para todas as instâncias do problema de otimização, produz uma solução cujo valor se aproxima do valor ótimo por um fator α .
- Classe de complexidade DTIME [Arora and Barak 2009]: Seja $T: \mathbb{R} \to \mathbb{R}$ uma função. Uma linguagem $L \in \mathbf{DTIME}(T(n))$ sse existe uma máquina de Turing que decide L em um tempo de execução de $c \cdot T(n)$, para uma constante c > 0.

Funções submodulares e Matroids • Vota

•
$$f: 2^X \to \mathbb{R}^+$$
 é submodular sse $f(A \cup B) + f(A \cap B) \le f(A) + f(B)$.

- Matroid:
 - Conjunto base X.
 - Família de elementos independentes $I \subseteq 2^X$.
 - Matroid é um par M = (X, I) tal que:
 - $\forall B \in I, A \subset B \Rightarrow A \in I$.
 - $\forall A, B \in I$; $|A| < |B| \Rightarrow \exists x \in B \setminus A$; $A \cup x \in I$.

- Seja S uma instância de SvMDST ($G = (V, E), T \subseteq V$ e $s \in T$). Seja $\Delta_{\mathcal{S}}^*$ uma solução ótima para \mathcal{S} .
- Instância \mathcal{D} de DSMDStP: $G = (V, E), s \in T, T_{\mathcal{D}} = T \setminus \{s\}$ e $k = \infty \left(\frac{\sum_{e \in E} C(e)}{\min_{e \in S} dist(e, t, G)} \right).$
- Seja $\Delta_{\mathcal{D}}^*$ uma solução ótima para \mathcal{D} .
- Para $k=\infty$, qualquer que seja a solução para S, ela satisfaz a
- Seja A uma algoritmo de aproximação por um fator α para o
- Então nós podemos aproximar Δ_s^* por um fator α .
- O fator de aproximação de Δ_S^* é > $(1 \varepsilon) \log_e |T|$ (Afirm. 1).

Impossibilidade de aproximação sublogarítmica



- Seja S uma instância de SvMDST ($G = (V, E), T \subseteq V$ e $s \in T$). Seja $\Delta_{\mathcal{S}}^*$ uma solução ótima para \mathcal{S} .
- Instância \mathcal{D} de DSMDStP: $G = (V, E), s \in T, T_{\mathcal{D}} = T \setminus \{s\}$ e $k = \infty \left(\frac{\sum_{e \in E} C(e)}{\min_{e \in V} \{ dist(s, t, G) \}} \right).$
- Seja $\Delta_{\mathcal{D}}^*$ uma solução ótima para \mathcal{D} .
- Para $k=\infty$, qualquer que seja a solução para S, ela satisfaz a
- Seja A uma algoritmo de aproximação por um fator α para o
- Então nós podemos aproximar Δ_s^* por um fator α .
- O fator de aproximação de Δ_S^* é > $(1 \varepsilon) \log_e |T|$ (Afirm. 1).

- Seja S uma instância de SvMDST ($G = (V, E), T \subseteq V$ e $s \in T$). Seja $\Delta_{\mathcal{S}}^*$ uma solução ótima para \mathcal{S} .
- Instância \mathcal{D} de DSMDStP: $G = (V, E), s \in T, T_{\mathcal{D}} = T \setminus \{s\}$ e $k = \infty \left(\frac{\sum_{e \in E} C(e)}{\min_{e \in V} \{ dist(s, t, G) \}} \right).$
- Seja Δ^{*}_D uma solução ótima para D.
- Para $k=\infty$, qualquer que seja a solução para S, ela satisfaz a
- Seja A uma algoritmo de aproximação por um fator α para o
- Então nós podemos aproximar Δ_s^* por um fator α .
- O fator de aproximação de Δ_s^* é $> (1 \varepsilon) \log_e |T|$ (Afirm. 1).

- Seja S uma instância de SvMDST ($G = (V, E), T \subseteq V$ e $s \in T$). Seja $\Delta_{\mathcal{S}}^*$ uma solução ótima para \mathcal{S} .
- Instância \mathcal{D} de DSMDStP: $G = (V, E), s \in T, T_{\mathcal{D}} = T \setminus \{s\}$ e $k = \infty \left(\frac{\sum_{e \in E} C(e)}{\min_{e \in V} \{ dist(s, t, G) \}} \right).$
- Seja Δ^{*}_D uma solução ótima para D.
- Para $k = \infty$, qualquer que seja a solução para S, ela satisfaz a restrição de spanner $\Longrightarrow \Delta_S^* = \Delta_D^*$.
- Seja A uma algoritmo de aproximação por um fator α para o
- Então nós podemos aproximar Δ_S^* por um fator α .
- O fator de aproximação de Δ_s^* é $> (1 \varepsilon) \log_e |T|$ (Afirm. 1).

- Seja S uma instância de SvMDST ($G = (V, E), T \subseteq V$ e $s \in T$). Seja $\Delta_{\mathcal{S}}^*$ uma solução ótima para \mathcal{S} .
- Instância \mathcal{D} de DSMDStP: $G = (V, E), s \in T, T_{\mathcal{D}} = T \setminus \{s\}$ e $k = \infty \left(\frac{\sum_{e \in E} C(e)}{\min_{e \in V} \{ dist(s, t, G) \}} \right).$
- Seja Δ^{*}_D uma solução ótima para D.
- Para $k = \infty$, qualquer que seja a solução para S, ela satisfaz a restrição de spanner $\Longrightarrow \Delta_S^* = \Delta_D^*$.
- Seja A uma algoritmo de aproximação por um fator α para o DSMDStP. Seja Δ^* o grau resultante.
- Então nós podemos aproximar Δ_S^* por um fator α .
- O fator de aproximação de Δ_s^* é $> (1 \varepsilon) \log_e |T|$ (Afirm. 1).

Impossibilidade de aproximação sublogarítmica



- Seja S uma instância de SvMDST ($G = (V, E), T \subseteq V$ e $s \in T$). Seja $\Delta_{\mathcal{S}}^*$ uma solução ótima para \mathcal{S} .
- Instância \mathcal{D} de DSMDStP: $G = (V, E), s \in T, T_{\mathcal{D}} = T \setminus \{s\}$ e $k = \infty \left(\frac{\sum_{e \in E} C(e)}{\min_{e \in V} \{ dist(s, t, G) \}} \right).$
- Seja Δ^{*}_D uma solução ótima para D.
- Para $k = \infty$, qualquer que seja a solução para S, ela satisfaz a restrição de spanner $\Longrightarrow \Delta_S^* = \Delta_D^*$.
- Seja A uma algoritmo de aproximação por um fator α para o DSMDStP. Seja Δ^* o grau resultante.
- Então nós podemos aproximar Δ_s^* por um fator α .
- O fator de aproximação de Δ_s^* é > $(1 \varepsilon) \log_e |T|$ (Afirm. 1).

Impossibilidade de aproximação sublogarítmica



- Seja S uma instância de SvMDST ($G = (V, E), T \subseteq V$ e $s \in T$). Seja $\Delta_{\mathcal{S}}^*$ uma solução ótima para \mathcal{S} .
- Instância \mathcal{D} de DSMDStP: $G = (V, E), s \in T, T_{\mathcal{D}} = T \setminus \{s\}$ e $k = \infty \left(\frac{\sum_{e \in E} C(e)}{\min_{e \in V} \{ dist(s, t, G) \}} \right).$
- Seja Δ^{*}_D uma solução ótima para D.
- Para $k = \infty$, qualquer que seja a solução para S, ela satisfaz a restrição de spanner $\Longrightarrow \Delta_S^* = \Delta_D^*$.
- Seja A uma algoritmo de aproximação por um fator α para o DSMDStP. Seja Δ^* o grau resultante.
- Então nós podemos aproximar Δ^{*}_S por um fator α.
- O fator de aproximação de Δ_S^* é > $(1 \varepsilon) \log_e |T|$ (Afirm. 1).

- Seja S uma instância de SvMDST ($G = (V, E), T \subseteq V$ e $s \in T$). Seja $\Delta_{\mathcal{S}}^*$ uma solução ótima para \mathcal{S} .
- Instância \mathcal{D} de DSMDStP: $G = (V, E), s \in T, T_{\mathcal{D}} = T \setminus \{s\}$ e $k = \infty \left(\frac{\sum_{e \in E} C(e)}{\min_{e \in V} \{ dist(s, t, G) \}} \right).$
- Seja Δ^{*}_D uma solução ótima para D.
- Para $k = \infty$, qualquer que seja a solução para S, ela satisfaz a restrição de spanner $\Longrightarrow \Delta_{\mathcal{S}}^* = \Delta_{\mathcal{D}}^*$.
- Seja A uma algoritmo de aproximação por um fator α para o DSMDStP. Seja Δ^* o grau resultante.
- Então nós podemos aproximar Δ^{*}_S por um fator α.
- O fator de aproximação de Δ_S^* é $> (1 \varepsilon) \log_e |T|$ (Afirm. 1). Então $(1 - \varepsilon) \log_e |T| < \frac{\Delta^*}{\Delta_c^*} \le \alpha \Longrightarrow \alpha > (1 - \varepsilon) \log_e |T|$.

Limite no grau máximo de $G_{\sqrt{I-Par}}$ (Algoritmo 2) •••••



- As árvores T_v computadas na linha 5 de Algoritmo 1 são disjuntas.
- Cada nó em T_{ν} possui grau máximo de \sqrt{I} .
- A_{Boots} é uma SPT de s para todos os nós em Roots e a
- $G_{\sqrt{I-Par}}$ corresponde à união de todas estas árvores (A_{Roots} e



Limite no grau máximo de $G_{\sqrt{l-Par}}$ (Algoritmo 2) • vota



- As árvores T_v computadas na linha 5 de Algoritmo 1 são disjuntas.
- Cada nó em T_{ν} possui grau máximo de \sqrt{I} .
- A_{Roots} é uma SPT de s para todos os nós em Roots e a
- $G_{\sqrt{I-Par}}$ corresponde à união de todas estas árvores (A_{Roots} e



Limite no grau máximo de $G_{\sqrt{l-Par}}$ (Algoritmo 2) • vota



- As árvores T_v computadas na linha 5 de Algoritmo 1 são disjuntas.
- Cada nó em T_{ν} possui grau máximo de \sqrt{I} .
- A_{Boots} é uma SPT de s para todos os nós em Roots e a cardinalidade máxima de *Roots* é de \sqrt{I} + 2 (Lema 2) \Rightarrow grau máximo de A_{Boots} é $\sqrt{I} + 2$.
- $G_{\sqrt{I-Par}}$ corresponde à união de todas estas árvores (A_{Roots} e



Limite no grau máximo de $G_{\sqrt{l}-Par}$ (Algoritmo 2) \bullet



- As árvores T_v computadas na linha 5 de Algoritmo 1 são disjuntas.
- Cada nó em T_{ν} possui grau máximo de \sqrt{I} .
- A_{Boots} é uma SPT de s para todos os nós em Roots e a cardinalidade máxima de *Roots* é de \sqrt{I} + 2 (Lema 2) \Rightarrow grau máximo de A_{Boots} é $\sqrt{I} + 2$.
- $G_{\sqrt{I}-Par}$ corresponde à união de todas estas árvores (A_{Roots} e T_{v}) \Longrightarrow grau máximo de qualquer nó é de $\sqrt{I} + \sqrt{I} + 2 = 2\sqrt{I} + 2$.





- Seja T* uma solução ótima para uma instância do DSMDStP.
- $\forall_{t \in UT}$ existe um caminho entre s e t em T^* . Seja $P_{T^*}(s,t)$ este caminho.
- Seja u o último nó de $P_{T^*}(s,t)$ que pertence a C. Seja v o próximo nó em $P_{T^*}(s,t)$.
- $x_{u,v}$ pertencerá a V_1 e $(x_{u,v},t)$ pertencerá a ε .
- Todos os nós em UT possuem esta propriedade e T* possui grau máximo d*.
 - \to Existe uma solução S^* t.q. \forall_u existe no máximo d^* pseudo nós $x_{u,v}$ em S^* , visto que $x_{u,v} \in S^*$ sse v é filho de u em T^* .
 - $\bullet \Rightarrow \max_{c \in C} \{ |S^* \cap A_c| \} \leq d^*.$





- ullet Seja T^* uma solução ótima para uma instância do DSMDStP.
- $\forall_{t \in UT}$ existe um caminho entre s e t em T^* . Seja $P_{T^*}(s,t)$ este caminho.
- Seja u o último nó de $P_{T^*}(s,t)$ que pertence a C. Seja v o próximo nó em $P_{T^*}(s,t)$.
- $x_{u,v}$ pertencerá a V_1 e $(x_{u,v},t)$ pertencerá a ε .
- Todos os nós em UT possuem esta propriedade e T^* possui grau máximo d^* .
 - \to Existe uma solução S^* t.q. \forall_u existe no máximo d^* pseudo nós $x_{u,v}$ em S^* , visto que $x_{u,v} \in S^*$ sse v é filho de u em T^* .
 - $\Rightarrow \max_{c \in C} \{ |S^* \cap A_c| \} \leq d^*.$





- ullet Seja T^* uma solução ótima para uma instância do DSMDStP.
- $\forall_{t \in UT}$ existe um caminho entre s e t em T^* . Seja $P_{T^*}(s,t)$ este caminho.
- Seja u o último nó de $P_{T^*}(s,t)$ que pertence a C. Seja v o próximo nó em $P_{T^*}(s,t)$.
- $x_{u,v}$ pertencerá a V_1 e $(x_{u,v},t)$ pertencerá a ε .
- Todos os nós em UT possuem esta propriedade e T^* possui grau máximo d^* .
 - \to Existe uma solução S^* t.q. \forall_u existe no máximo d^* pseudo nós $x_{u,v}$ em S^* , visto que $x_{u,v} \in S^*$ sse v é filho de u em T^* .
 - $\bullet \Rightarrow \max_{c \in C} \{ |S^* \cap A_c| \} \leq d^*.$



- Seja T* uma solução ótima para uma instância do DSMDStP.
- $\forall_{t \in UT}$ existe um caminho entre s e t em T^* . Seja $P_{T^*}(s,t)$ este caminho.
- Seja u o último nó de $P_{T^*}(s,t)$ que pertence a C. Seja v o próximo nó em $P_{T^*}(s,t)$.
- $x_{u,v}$ pertencerá a V_1 e $(x_{u,v},t)$ pertencerá a ε .
- Todos os nós em UT possuem esta propriedade e T* possui grau máximo d*.
 - \to Existe uma solução S^* t.q. \forall_u existe no máximo d^* pseudo nós $x_{u,v}$ em S^* , visto que $x_{u,v} \in S^*$ sse v é filho de u em T^* .
 - $\bullet \Rightarrow \max_{c \in C} \{ |S^* \cap A_c| \} \leq d^*.$



- ullet Seja T^* uma solução ótima para uma instância do DSMDStP.
- $\forall_{t \in UT}$ existe um caminho entre s e t em T^* . Seja $P_{T^*}(s,t)$ este caminho.
- Seja u o último nó de $P_{T^*}(s,t)$ que pertence a C. Seja v o próximo nó em $P_{T^*}(s,t)$.
- $x_{u,v}$ pertencerá a V_1 e $(x_{u,v},t)$ pertencerá a ε .
- Todos os nós em UT possuem esta propriedade e T* possui grau máximo d*.
 - \to Existe uma solução S^* t.q. \forall_u existe no máximo d^* pseudo nós $x_{u,v}$ em S^* , visto que $x_{u,v} \in S^*$ sse v é filho de u em T^* .
 - $\bullet \Rightarrow \max_{c \in C} \{ |S^* \cap A_c| \} \leq d^*.$



Limite da interseção entre a solução do MSC e cada partição de V_1

- $d^* \ge val(S^*)$ para uma solução ótima S^* de β .
- $|V_2| \le 1$.
- O lema é direto.

Limite da interseção entre a solução do MSC e cada partição de V_1

- $d^* \ge val(S^*)$ para uma solução ótima S^* de β .
- $|V_2| \leq I$.
- O lema é direto

Limite da interseção entre a solução do MSC e cada partição de V_1 \bullet vota

- $d^* \ge val(S^*)$ para uma solução ótima S^* de β .
- $|V_2| \leq I$.
- O lema é direto



Limite da interseção entre a solução do MSC e cada partição de V_1 • votar

- $d^* \ge val(S^*)$ para uma solução ótima S^* de β .
- $|V_2| \leq I$.
- O lema é direto.

Grau máximo de G_f vota

- Nós são adicionados nas duas fases.
- G_f tem grau máximo de $2\sqrt{I} + 2$ ao final da primeira fase (Lema 3).
- Novos nós adicionados (fase 2) tem no máximo grau \sqrt{I} (ausência de \sqrt{I} -bad nodes).
- Nó u é selecionado apenas se $x_{u,v} \in D$.
- Nó u pode ter seu grau incrementado por no máximo O(log l) · d* (Lema 5).
- Provado o limite de $< 2\sqrt{I} + 2 + O(\log I) \cdot d^*$.

Grau máximo de G_f vota

- Nós são adicionados nas duas fases.
- G_f tem grau máximo de $2\sqrt{I} + 2$ ao final da primeira fase (Lema 3).
- Novos nós adicionados (fase 2) tem no máximo grau \sqrt{I} (ausência de \sqrt{I} -bad nodes).
- Nó u é selecionado apenas se $x_{u,v} \in D$.
- Nó u pode ter seu grau incrementado por no máximo O(log l) · d* (Lema 5).
- Provado o limite de $< 2\sqrt{I} + 2 + O(\log I) \cdot d^*$.

Grau máximo de G_f vota

- Nós são adicionados nas duas fases.
- G_f tem grau máximo de $2\sqrt{I} + 2$ ao final da primeira fase (Lema 3).
- Novos nós adicionados (fase 2) tem no máximo grau \sqrt{I} (ausência de \sqrt{I} -bad nodes).
- Nó u é selecionado apenas se $x_{u,v} \in D$.
- Nó u pode ter seu grau incrementado por no máximo O(log l) · d* (Lema 5).
- Provado o limite de $< 2\sqrt{I} + 2 + O(\log I) \cdot d^*$.

Grau máximo de G_f

- Nós são adicionados nas duas fases.
- G_f tem grau máximo de $2\sqrt{I} + 2$ ao final da primeira fase (Lema 3).
- Novos nós adicionados (fase 2) tem no máximo grau \sqrt{I} (ausência de \sqrt{I} -bad nodes).
- Nó u é selecionado apenas se $x_{u,v} \in D$.
- Nó u pode ter seu grau incrementado por no máximo O(log l) · d* (Lema 5).
- Provado o limite de $< 2\sqrt{I} + 2 + O(\log I) \cdot d^*$.

Grau máximo de G_f voltar

- Nós são adicionados nas duas fases.
- G_f tem grau máximo de $2\sqrt{I} + 2$ ao final da primeira fase (Lema 3).
- Novos nós adicionados (fase 2) tem no máximo grau \sqrt{I} (ausência de \sqrt{I} -bad nodes).
- Nó u é selecionado apenas se $x_{u,v} \in D$.
- Nó u pode ter seu grau incrementado por no máximo O(log l) · d* (Lema 5).
- Provado o limite de $\leq 2\sqrt{l} + 2 + O(\log l) \cdot d^*$.

Grau máximo de G_f

- Nós são adicionados nas duas fases.
- G_f tem grau máximo de $2\sqrt{I} + 2$ ao final da primeira fase (Lema 3).
- Novos nós adicionados (fase 2) tem no máximo grau \sqrt{I} (ausência de \sqrt{I} -bad nodes).
- Nó u é selecionado apenas se $x_{u,v} \in D$.
- Nó u pode ter seu grau incrementado por no máximo O(log l) · d* (Lema 5).
- Provado o limite de $\leq 2\sqrt{I} + 2 + O(\log I) \cdot d^*$.



- Se t é coberto na primeira fase, $dist(s, t, G_t) \le k \cdot dist(s, t, G)$.
- Todos os nós v cobertos na fase 1 fazem parte de um caminho
- $\forall_{t \in T}$ cobertos apenas na fase 2, $x_{u,v}$ e $(x_{u,v},t)$ são inseridas na
- Em particular, $C(u, v) + dist(v, t, G(U)) \le k \cdot dist(s, t, G)$.
- Como $dist(s, u, G) \leq k \cdot dist(s, t_{max}, G) \Rightarrow dist(s, t, G_t) \leq$





- Se t é coberto na primeira fase, $dist(s, t, G_t) \leq k \cdot dist(s, t, G)$.
- Todos os nós v cobertos na fase 1 fazem parte de um caminho entre s e um t t.q. seu custo é $\leq k \cdot dist(s, t, G) \rightarrow \leq k \cdot dist(s, t_{max}, G).$
- $\forall_{t \in T}$ cobertos apenas na fase 2, $x_{u,v}$ e $(x_{u,v}, t)$ são inseridas na
- Em particular, $C(u, v) + dist(v, t, G(U)) \le k \cdot dist(s, t, G)$.
- Como $dist(s, u, G) \leq k \cdot dist(s, t_{max}, G) \Rightarrow dist(s, t, G_t) \leq$



- Se t é coberto na primeira fase, $dist(s, t, G_t) \leq k \cdot dist(s, t, G)$.
- Todos os nós v cobertos na fase 1 fazem parte de um caminho entre s e um t t.q. seu custo é $\leq k \cdot dist(s, t, G) \rightarrow \leq k \cdot dist(s, t_{max}, G).$
- $\forall_{t \in T}$ cobertos apenas na fase 2, $x_{u,v}$ e $(x_{u,v},t)$ são inseridas na instância do MSC se $dist(s, u, G) + C(u, v) + dist(v, t, G(U)) \le k \cdot dist(s, t, G).$
- Em particular, $C(u, v) + dist(v, t, G(U)) \le k \cdot dist(s, t, G)$.
- Como $dist(s, u, G) \leq k \cdot dist(s, t_{max}, G) \Rightarrow dist(s, t, G_t) \leq$



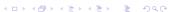


- Se t é coberto na primeira fase, $dist(s, t, G_t) \leq k \cdot dist(s, t, G)$.
- Todos os nós v cobertos na fase 1 fazem parte de um caminho entre s e um t t.q. seu custo é $\leq k \cdot dist(s, t, G) \rightarrow \leq k \cdot dist(s, t_{max}, G).$
- $\forall_{t \in T}$ cobertos apenas na fase 2, $x_{u,v}$ e $(x_{u,v},t)$ são inseridas na instância do MSC se $dist(s, u, G) + C(u, v) + dist(v, t, G(U)) \le k \cdot dist(s, t, G).$
- Em particular, $C(u, v) + dist(v, t, G(U)) \le k \cdot dist(s, t, G)$.
- Como $dist(s, u, G) \le k \cdot dist(s, t_{max}, G) \Rightarrow dist(s, t, G_f) \le dist(s, G_f) \le dis$





- Se t é coberto na primeira fase, $dist(s, t, G_t) \leq k \cdot dist(s, t, G)$.
- Todos os nós v cobertos na fase 1 fazem parte de um caminho entre s e um t t.q. seu custo é $\leq k \cdot dist(s, t, G) \rightarrow \leq k \cdot dist(s, t_{max}, G).$
- $\forall_{t \in T}$ cobertos apenas na fase 2, $x_{u,v}$ e $(x_{u,v},t)$ são inseridas na instância do MSC se $dist(s, u, G) + C(u, v) + dist(v, t, G(U)) \le k \cdot dist(s, t, G).$
- Em particular, $C(u, v) + dist(v, t, G(U)) \le k \cdot dist(s, t, G)$.
- Como $dist(s, u, G) \le k \cdot dist(s, t_{max}, G) \Rightarrow dist(s, t, G_f) \le dist(s, G_f) \le$ $k \cdot (dist(s, t, G) + \cdot dist(s, t_{max}, G)).$



Teorema Principal Volta

- A_f é gerada computando caminhos de custo mínimo em G_f entre s e t ∈ T.
- O teorema segue do resultado de grau máximo (Lema 6) e do resultado de custo máximo (Lema 7).



Teorema Principal Voltar

- A_f é gerada computando caminhos de custo mínimo em G_f entre s e t ∈ T.
- O teorema segue do resultado de grau máximo (Lema 6) e do resultado de custo máximo (Lema 7).

Teorema Principal **Protection**

- A_f é gerada computando caminhos de custo mínimo em G_f entre s e t ∈ T.
- O teorema segue do resultado de grau máximo (Lema 6) e do resultado de custo máximo (Lema 7).





- Seja T* uma solução ótima para uma instância de DSMDStP.
- Considere u e v os nós no caminho entre s e t como mencionados na prova do Lema 4.
- Então, existirá um $x_{u,v}$ em V_1 e uma aresta $(x_{u,v},t)$ em ε (idependente da iteração).
- Após aplicar algoritmo de [Chekuri and Kumar 2004] (linha 8), pelo menos um caminho existirá entre u e t.
 - → Pelo menos um caminho será escolhido por SIM para fazer parte de A_I (linha 11).





- Seja T* uma solução ótima para uma instância de DSMDStP.
- Considere u e v os nós no caminho entre s e t como mencionados na prova do Lema 4.
- Então, existirá um $x_{u,v}$ em V_1 e uma aresta $(x_{u,v},t)$ em ε (idependente da iteração).
- Após aplicar algoritmo de [Chekuri and Kumar 2004] (linha 8), pelo menos um caminho existirá entre u e t.
 - → Pelo menos um caminho será escolhido por SIM para fazer parte de A_i (linha 11).





- Seja T* uma solução ótima para uma instância de DSMDStP.
- Considere u e v os nós no caminho entre s e t como mencionados na prova do Lema 4.
- Então, existirá um $x_{u,v}$ em V_1 e uma aresta $(x_{u,v},t)$ em ε (idependente da iteração).
- Após aplicar algoritmo de [Chekuri and Kumar 2004] (linha 8), pelo menos um caminho existirá entre u e t.
 - → Pelo menos um caminho será escolhido por SIM para fazer parte de A₁ (linha 11).



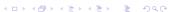


- Seja T* uma solução ótima para uma instância de DSMDStP.
- Considere u e v os nós no caminho entre s e t como mencionados na prova do Lema 4.
- Então, existirá um $x_{u,v}$ em V_1 e uma aresta $(x_{u,v},t)$ em ε (idependente da iteração).
- Após aplicar algoritmo de [Chekuri and Kumar 2004] (linha 8), pelo menos um caminho existirá entre *u* e *t*.





- Seja T* uma solução ótima para uma instância de DSMDStP.
- Considere u e v os nós no caminho entre s e t como mencionados na prova do Lema 4.
- Então, existirá um $x_{u,v}$ em V_1 e uma aresta $(x_{u,v},t)$ em ε (idependente da iteração).
- Após aplicar algoritmo de [Chekuri and Kumar 2004] (linha 8), pelo menos um caminho existirá entre u e t.
 - → Pelo menos um caminho será escolhido por SIM para fazer parte de A_f (linha 11).





- Seja T* uma solução ótima para uma instância de DSMDStP.
- Considere u e v os nós no caminho entre s e t como mencionados na prova do Lema 4.
- Então, existirá um $x_{u,v}$ em V_1 e uma aresta $(x_{u,v},t)$ em ε (idependente da iteração).
- Após aplicar algoritmo de [Chekuri and Kumar 2004] (linha 8), pelo menos um caminho existirá entre u e t.
 - \rightarrow Pelo menos um caminho será escolhido por SIM para fazer parte de \mathcal{A}_f (linha 11).





- Considere o número de terminais I.
- Se / é um Quadrado Perfeito (QP), todos os terminais serão cobertos em \sqrt{I} iterações ($|\sqrt{I}| = \sqrt{I}$).
- Considere / não sendo um quadrado perfeito.
- Seja $\sqrt{I} = \lfloor \sqrt{I} \rfloor + \mathcal{F}$, onde $\mathcal{F} \in]0, 1[$. Então $I = (\lfloor \sqrt{I} \rfloor + \mathcal{F}) \cdot (\lfloor \sqrt{I} \rfloor + \mathcal{F}).$ • $\rightarrow I = \lfloor \sqrt{I} \rfloor \cdot (\lfloor \sqrt{I} \rfloor + \lfloor (2 \cdot \mathcal{F}) + (\mathcal{F} \cdot \frac{\mathcal{F}}{\lfloor \sqrt{I} \rfloor}) \rfloor)$
- Podemos dividir I em dois números: $\lfloor \sqrt{I} \rfloor$ e $(\lfloor \sqrt{I} \rfloor + \lfloor (2 \cdot \mathcal{F}) + (\mathcal{F} \cdot \frac{\mathcal{F}}{\lfloor \sqrt{I} \rfloor}) \rfloor)$.
 - [√l]: coincide com o número de terminais cobertos em cada iteração.
 - Precisamos encontrar um inteiro I_{UB} que representa um limite para o número real $(|\sqrt{I}| + [(2 \cdot \mathcal{F}) + (\mathcal{F} \cdot \frac{\mathcal{F}}{2})])$.

- Considere o número de terminais I.
- Se / é um Quadrado Perfeito (QP), todos os terminais serão cobertos em \sqrt{I} iterações ($\lfloor \sqrt{I} \rfloor = \sqrt{I}$).
- Considere / não sendo um quadrado perfeito.
- Seja $\sqrt{I} = \lfloor \sqrt{I} \rfloor + \mathcal{F}$, onde $\mathcal{F} \in]0, 1[$. Então $I = (\lfloor \sqrt{I} \rfloor + \mathcal{F}) \cdot (\lfloor \sqrt{I} \rfloor + \mathcal{F}).$
 - $\bullet \to I = \lfloor \sqrt{I} \rfloor \cdot \left(\lfloor \sqrt{I} \rfloor + \lfloor (2 \cdot \mathcal{F}) + \left(\mathcal{F} \cdot \frac{\mathcal{F}}{\lfloor \sqrt{I} \rfloor} \right) \rfloor \right)$
- Podemos dividir I em dois números: $\lfloor \sqrt{I} \rfloor$ e $(\lfloor \sqrt{I} \rfloor + \lfloor (2 \cdot \mathcal{F}) + (\mathcal{F} \cdot \frac{\mathcal{F}}{\lfloor \sqrt{I} \rfloor}) \rfloor)$.
 - [√/]: coincide com o número de terminais cobertos em cada iteracão.
 - Precisamos encontrar um inteiro I_{UB} que representa um limite para o número real $(|\sqrt{I}| + [(2 \cdot \mathcal{F}) + (\mathcal{F} \cdot \frac{\mathcal{F}}{\mathcal{F}})])$.

- Considere o número de terminais I.
- Se / é um Quadrado Perfeito (QP), todos os terminais serão cobertos em \sqrt{I} iterações ($|\sqrt{I}| = \sqrt{I}$).
- Considere / não sendo um quadrado perfeito.

• Seja
$$\sqrt{I} = \lfloor \sqrt{I} \rfloor + \mathcal{F}$$
, onde $\mathcal{F} \in]0,1[$. Então $I = (\lfloor \sqrt{I} \rfloor + \mathcal{F}) \cdot (\lfloor \sqrt{I} \rfloor + \mathcal{F}).$

- Podemos dividir I em dois números: $\lfloor \sqrt{I} \rfloor$ e $(\lfloor \sqrt{I} \rfloor + \lfloor (2 \cdot \mathcal{F}) + (\mathcal{F} \cdot \frac{\mathcal{F}}{\lfloor \sqrt{I} \rfloor}) \rfloor)$.
 - iteração.

 Precisamos encontrar um inteiro l_{UB} que representa um limite para

- Considere o número de terminais I.
- Se / é um Quadrado Perfeito (QP), todos os terminais serão cobertos em \sqrt{I} iterações ($\lfloor \sqrt{I} \rfloor = \sqrt{I}$).
- Considere / não sendo um quadrado perfeito.
- Seja $\sqrt{I} = \lfloor \sqrt{I} \rfloor + \mathcal{F}$, onde $\mathcal{F} \in]0, 1[$. Então $I = (\lfloor \sqrt{I} \rfloor + \mathcal{F}) \cdot (\lfloor \sqrt{I} \rfloor + \mathcal{F}).$

$$\bullet \to I = \lfloor \sqrt{I} \rfloor \cdot \left(\lfloor \sqrt{I} \rfloor + \left[(2 \cdot \mathcal{F}) + \left(\mathcal{F} \cdot \frac{\mathcal{F}}{\lfloor \sqrt{I} \rfloor} \right) \right] \right).$$

- Podemos dividir I em dois números: $\lfloor \sqrt{I} \rfloor$ e $(\lfloor \sqrt{I} \rfloor + \lfloor (2 \cdot \mathcal{F}) + (\mathcal{F} \cdot \frac{\mathcal{F}}{\lfloor \sqrt{I} \rfloor}) \rfloor)$.
 - | √I|: coincide com o número de terminais cobertos em cada iteracão.
 - Precisamos encontrar um inteiro I_{UB} que representa um limite para o número real $(|\sqrt{I}| + [(2 \cdot \mathcal{F}) + (\mathcal{F} \cdot \frac{\mathcal{F}}{2})])$.

Demonstração.

- Considere o número de terminais I.
- Se / é um Quadrado Perfeito (QP), todos os terminais serão cobertos em \sqrt{I} iterações ($\lfloor \sqrt{I} \rfloor = \sqrt{I}$).
- Considere / não sendo um quadrado perfeito.
- Seja $\sqrt{I} = \lfloor \sqrt{I} \rfloor + \mathcal{F}$, onde $\mathcal{F} \in]0, 1[$. Então $I = (\lfloor \sqrt{I} \rfloor + \mathcal{F}) \cdot (\lfloor \sqrt{I} \rfloor + \mathcal{F}).$

$$\bullet \to I = \lfloor \sqrt{I} \rfloor \cdot \left(\lfloor \sqrt{I} \rfloor + \left[(2 \cdot \mathcal{F}) + \left(\mathcal{F} \cdot \frac{\mathcal{F}}{\lfloor \sqrt{I} \rfloor} \right) \right] \right).$$

• Podemos dividir I em dois números: $\lfloor \sqrt{I} \rfloor$ e $(\lfloor \sqrt{I} \rfloor + \lfloor (2 \cdot \mathcal{F}) + (\mathcal{F} \cdot \frac{\mathcal{F}}{\lfloor \sqrt{I} \rfloor}) \rfloor)$.

- [√/]: coincide com o número de terminais cobertos em cada iteracão.
- Precisamos encontrar um inteiro I_{UB} que representa um limite para o número real $(|\sqrt{l}| + [(2 \cdot \mathcal{F}) + (\mathcal{F} \cdot \frac{\mathcal{F}}{2})])$.

- Considere o número de terminais I.
- Se / é um Quadrado Perfeito (QP), todos os terminais serão cobertos em \sqrt{I} iterações ($\lfloor \sqrt{I} \rfloor = \sqrt{I}$).
- Considere / não sendo um quadrado perfeito.
- Seja $\sqrt{I} = \lfloor \sqrt{I} \rfloor + \mathcal{F}$, onde $\mathcal{F} \in]0,1[$. Então $I = (\lfloor \sqrt{I} \rfloor + \mathcal{F}) \cdot (\lfloor \sqrt{I} \rfloor + \mathcal{F}).$
 - $\bullet \to I = \lfloor \sqrt{I} \rfloor \cdot \left(\lfloor \sqrt{I} \rfloor + \left[(2 \cdot \mathcal{F}) + \left(\mathcal{F} \cdot \frac{\mathcal{F}}{\lfloor \sqrt{I} \rfloor} \right) \right] \right).$
- Podemos dividir I em dois números: $\lfloor \sqrt{I} \rfloor$ e $(\lfloor \sqrt{I} \rfloor + [(2 \cdot \mathcal{F}) + (\mathcal{F} \cdot \frac{\mathcal{F}}{\lfloor \sqrt{I} \rfloor})])$.
 - L√I]: coincide com o número de terminais cobertos em cada iteracão.
 - Precisamos encontrar um inteiro I_{UB} que representa um limite para o número real $(|\sqrt{I}| + [(2 \cdot \mathcal{F}) + (\mathcal{F} \cdot \frac{\mathcal{F}}{\sqrt{G}})])$.
 - I_{IB} iterações é suficiente para cobrir / terminais.

- Considere o número de terminais I.
- Se / é um Quadrado Perfeito (QP), todos os terminais serão cobertos em \sqrt{I} iterações ($\lfloor \sqrt{I} \rfloor = \sqrt{I}$).
- Considere / não sendo um quadrado perfeito.
- Seja $\sqrt{I} = \lfloor \sqrt{I} \rfloor + \mathcal{F}$, onde $\mathcal{F} \in]0, 1[$. Então $I = (\lfloor \sqrt{I} \rfloor + \mathcal{F}) \cdot (\lfloor \sqrt{I} \rfloor + \mathcal{F}).$
 - $\bullet \to I = \lfloor \sqrt{I} \rfloor \cdot \left(\lfloor \sqrt{I} \rfloor + \left[(2 \cdot \mathcal{F}) + \left(\mathcal{F} \cdot \frac{\mathcal{F}}{\lfloor \sqrt{I} \rfloor} \right) \right] \right).$
- Podemos dividir I em dois números: $\lfloor \sqrt{I} \rfloor$ e $(\lfloor \sqrt{I} \rfloor + [(2 \cdot \mathcal{F}) + (\mathcal{F} \cdot \frac{\mathcal{F}}{\lfloor \sqrt{I} \rfloor})])$.
 - $\lfloor \sqrt{I} \rfloor$: coincide com o número de terminais cobertos em cada iteração.
 - Precisamos encontrar um inteiro I_{UB} que representa um limite para o número real $(|\sqrt{I}| + [(2 \cdot \mathcal{F}) + (\mathcal{F} \cdot \frac{\mathcal{F}}{|\mathcal{F}|})])$.
 - I_{UB} iterações é suficiente para cobrir / terminais.

- Considere o número de terminais I.
- Se / é um Quadrado Perfeito (QP), todos os terminais serão cobertos em \sqrt{I} iterações ($\lfloor \sqrt{I} \rfloor = \sqrt{I}$).
- Considere / não sendo um quadrado perfeito.
- Seja $\sqrt{I} = \lfloor \sqrt{I} \rfloor + \mathcal{F}$, onde $\mathcal{F} \in]0,1[$. Então $I = (\lfloor \sqrt{I} \rfloor + \mathcal{F}) \cdot (\lfloor \sqrt{I} \rfloor + \mathcal{F}).$
 - $\bullet \to I = \lfloor \sqrt{I} \rfloor \cdot \left(\lfloor \sqrt{I} \rfloor + \left[(2 \cdot \mathcal{F}) + \left(\mathcal{F} \cdot \frac{\mathcal{F}}{\lfloor \sqrt{I} \rfloor} \right) \right] \right).$
- Podemos dividir I em dois números: $\lfloor \sqrt{I} \rfloor$ e $(\lfloor \sqrt{I} \rfloor + [(2 \cdot \mathcal{F}) + (\mathcal{F} \cdot \frac{\mathcal{F}}{\lfloor \sqrt{I} \rfloor})])$.
 - L√I]: coincide com o número de terminais cobertos em cada iteracão.
 - Precisamos encontrar um inteiro I_{UB} que representa um limite para o número real $(\lfloor \sqrt{I} \rfloor + [(2 \cdot \mathcal{F}) + (\mathcal{F} \cdot \frac{\mathcal{F}}{\lfloor \sqrt{I} \rfloor})])$.
 - I_{UB} iterações é suficiente para cobrir I terminais

- Considere o número de terminais I.
- Se / é um Quadrado Perfeito (QP), todos os terminais serão cobertos em \sqrt{I} iterações ($|\sqrt{I}| = \sqrt{I}$).
- Considere / não sendo um quadrado perfeito.
- Seja $\sqrt{I} = |\sqrt{I}| + \mathcal{F}$, onde $\mathcal{F} \in]0, 1[$. Então $I = (|\sqrt{I}| + \mathcal{F}) \cdot (|\sqrt{I}| + \mathcal{F}).$
 - $\bullet \to I = \lfloor \sqrt{I} \rfloor \cdot \left(\lfloor \sqrt{I} \rfloor + \left\lceil (2 \cdot \mathcal{F}) + \left(\mathcal{F} \cdot \frac{\mathcal{F}}{\lfloor \sqrt{I} \rfloor} \right) \right\rceil \right).$
- Podemos dividir *I* em dois números: $|\sqrt{I}|$ e $(\lfloor \sqrt{I} \rfloor + [(2 \cdot \mathcal{F}) + (\mathcal{F} \cdot \frac{\mathcal{F}}{\lfloor \sqrt{I} \rfloor})]).$
 - $|\sqrt{I}|$: coincide com o número de terminais cobertos em cada iteração.
 - Precisamos encontrar um inteiro I_{UB} que representa um limite para o número real $(\lfloor \sqrt{I} \rfloor + [(2 \cdot \mathcal{F}) + (\mathcal{F} \cdot \frac{\mathcal{F}}{\lfloor \sqrt{I} \rfloor})])$.
 - I_{UB} iterações é suficiente para cobrir / terminais.



Número de iterações do SIM •votar

•
$$\lfloor \sqrt{I} \rfloor + (2 \cdot \mathcal{F}) + (\mathcal{F} \cdot \frac{\mathcal{F}}{\lfloor \sqrt{I} \rfloor}) = \lfloor \sqrt{I} \rfloor + \frac{I - \lfloor \sqrt{I} \rfloor^2}{\lfloor \sqrt{I} \rfloor}$$
.

- Como $\lfloor \sqrt{I} \rfloor^2$ é o maior quadrado perfeito menor que $I \to (I \lfloor \sqrt{I} \rfloor^2) \le 2 \cdot \lfloor \sqrt{I} \rfloor$ (prova adiada).
 - $\bullet \to \lfloor \sqrt{I} \rfloor + \frac{I \lfloor \sqrt{I} \rfloor}{\lfloor \sqrt{I} \rfloor} \le \lfloor \sqrt{I} \rfloor + \frac{2 \lfloor \sqrt{I} \rfloor}{\lfloor \sqrt{I} \rfloor} = \lfloor \sqrt{I} \rfloor + 2.$
- Provar que $(I \lfloor \sqrt{I} \rfloor^2) \le 2 \cdot \lfloor \sqrt{I} \rfloor$ (quadrados perfeitos).
 - Seja SR_y um quadrado perfeito e R_y sua raiz
 - Para dois quadrados perfeitos consecutivos SR_i e
 SR. SR. R. I. R.
 - Em outras palavras: $SR_i SR_i = 2 \cdot R_i + 1$
 - Seja X um quadrado n\u00e3o-perfeito e seja GSR_X o maior quadrado perfeito menor que X. Seja R_X a raiz de GSR_X.

Número de iterações do SIM •vota

Demonstração.

•
$$\lfloor \sqrt{I} \rfloor + (2 \cdot \mathcal{F}) + (\mathcal{F} \cdot \frac{\mathcal{F}}{\lfloor \sqrt{I} \rfloor}) = \lfloor \sqrt{I} \rfloor + \frac{I - \lfloor \sqrt{I} \rfloor^2}{\lfloor \sqrt{I} \rfloor}$$
.

• Como $\lfloor \sqrt{I} \rfloor^2$ é o maior quadrado perfeito menor que $I \to (I - \lfloor \sqrt{I} \rfloor^2) \le 2 \cdot \lfloor \sqrt{I} \rfloor$ (prova adiada).

$$\bullet \to \lfloor \sqrt{I} \rfloor + \frac{I - \lfloor \sqrt{I} \rfloor^2}{\lfloor \sqrt{I} \rfloor} \le \lfloor \sqrt{I} \rfloor + \frac{2 \cdot \lfloor \sqrt{I} \rfloor}{\lfloor \sqrt{I} \rfloor} = \lfloor \sqrt{I} \rfloor + 2.$$

- Provar que $(I \lfloor \sqrt{I} \rfloor^2) \le 2 \cdot \lfloor \sqrt{I} \rfloor$ (quadrados perfeitos).
 - ullet Seja SR_X um quadrado perfeito e R_X sua raiz.
 - Para dois quadrados perfeitos consecutivos SR_i e
 - $SR_j o SR_j SR_i = R_i + R_j$.
 - Em outras palavras: $SR_i SR_i = 2 \cdot R_i + 1$.
 - Seja X um quadrado não-perfeito e seja GSR_X o maior quadrado perfeito menor que X. Seja R_Y a raiz de GSR_Y.

Número de iterações do SIM •••••

Demonstração.

•
$$\lfloor \sqrt{I} \rfloor + (2 \cdot \mathcal{F}) + (\mathcal{F} \cdot \frac{\mathcal{F}}{\lfloor \sqrt{I} \rfloor}) = \lfloor \sqrt{I} \rfloor + \frac{I - \lfloor \sqrt{I} \rfloor^2}{\lfloor \sqrt{I} \rfloor}$$
.

• Como $\lfloor \sqrt{I} \rfloor^2$ é o maior quadrado perfeito menor que $I \to (I - \lfloor \sqrt{I} \rfloor^2) \le 2 \cdot \lfloor \sqrt{I} \rfloor$ (prova adiada).

$$\bullet \to \lfloor \sqrt{I} \rfloor + \frac{I - \lfloor \sqrt{I} \rfloor^2}{\lfloor \sqrt{I} \rfloor} \le \lfloor \sqrt{I} \rfloor + \frac{2 \cdot \lfloor \sqrt{I} \rfloor}{\lfloor \sqrt{I} \rfloor} = \lfloor \sqrt{I} \rfloor + 2.$$

• Provar que $(I - \lfloor \sqrt{I} \rfloor^2) \le 2 \cdot \lfloor \sqrt{I} \rfloor$ (quadrados perfeitos).

Demonstração.

•
$$\lfloor \sqrt{I} \rfloor + (2 \cdot \mathcal{F}) + (\mathcal{F} \cdot \frac{\mathcal{F}}{\lfloor \sqrt{I} \rfloor}) = \lfloor \sqrt{I} \rfloor + \frac{I - \lfloor \sqrt{I} \rfloor^2}{\lfloor \sqrt{I} \rfloor}$$
.

• Como $\lfloor \sqrt{I} \rfloor^2$ é o maior quadrado perfeito menor que $I \to (I - \lfloor \sqrt{I} \rfloor^2) \le 2 \cdot \lfloor \sqrt{I} \rfloor$ (prova adiada).

$$\bullet \to \lfloor \sqrt{I} \rfloor + \frac{I - \lfloor \sqrt{I} \rfloor^2}{\lfloor \sqrt{I} \rfloor} \le \lfloor \sqrt{I} \rfloor + \frac{2 \cdot \lfloor \sqrt{I} \rfloor}{\lfloor \sqrt{I} \rfloor} = \lfloor \sqrt{I} \rfloor + 2.$$

- Provar que $(I \lfloor \sqrt{I} \rfloor^2) \le 2 \cdot \lfloor \sqrt{I} \rfloor$ (quadrados perfeitos).
 - Seja SR_x um quadrado perfeito e R_x sua raiz
 - Para dois quadrados perfeitos consecutivos SR_i e
 SR_i = SR_i = R_i + R_i
 - Em outras palavras: $SR_i SR_i = 2 \cdot R_i + 1$
 - Seja X um quadrado n\u00e3o-perfeito e seja GSR_X o maior quadrado perfeito menor que X. Seja R_X a raiz de GSR_X.

• $X - GSR_X \leq 2 \cdot R_X$.

Demonstração.

•
$$\lfloor \sqrt{I} \rfloor + (2 \cdot \mathcal{F}) + (\mathcal{F} \cdot \frac{\mathcal{F}}{\lfloor \sqrt{I} \rfloor}) = \lfloor \sqrt{I} \rfloor + \frac{I - \lfloor \sqrt{I} \rfloor^2}{\lfloor \sqrt{I} \rfloor}$$
.

• Como $\lfloor \sqrt{I} \rfloor^2$ é o maior quadrado perfeito menor que $I \to (I - \lfloor \sqrt{I} \rfloor^2) \le 2 \cdot \lfloor \sqrt{I} \rfloor$ (prova adiada).

$$\bullet \to \lfloor \sqrt{I} \rfloor + \frac{I - \lfloor \sqrt{I} \rfloor^2}{\lfloor \sqrt{I} \rfloor} \le \lfloor \sqrt{I} \rfloor + \frac{2 \cdot \lfloor \sqrt{I} \rfloor}{\lfloor \sqrt{I} \rfloor} = \lfloor \sqrt{I} \rfloor + 2.$$

- Provar que $(I \lfloor \sqrt{I} \rfloor^2) \le 2 \cdot \lfloor \sqrt{I} \rfloor$ (quadrados perfeitos).
 - Seja SR_x um quadrado perfeito e R_x sua raiz.
 - Para dois quadrados perfeitos consecutivos SR_i e
 SR_i → SR_i − SR_i = R_i + R_i
 - Em outras palavras: $SR_i SR_i = 2 \cdot R_i + 1$
 - Seja X um quadrado n\u00e3o-perfeito e seja GSR_X o maior quadrado perfeito menor que X. Seja R_X a raiz de GSR_X.

• $X - GSR_X < 2 \cdot R_X$.

Demonstração.

•
$$\lfloor \sqrt{I} \rfloor + (2 \cdot \mathcal{F}) + (\mathcal{F} \cdot \frac{\mathcal{F}}{\lfloor \sqrt{I} \rfloor}) = \lfloor \sqrt{I} \rfloor + \frac{I - \lfloor \sqrt{I} \rfloor^2}{\lfloor \sqrt{I} \rfloor}$$
.

• Como $\lfloor \sqrt{I} \rfloor^2$ é o maior quadrado perfeito menor que $I \to (I - \lfloor \sqrt{I} \rfloor^2) \le 2 \cdot \lfloor \sqrt{I} \rfloor$ (prova adiada).

$$\bullet \to \lfloor \sqrt{I} \rfloor + \frac{I - \lfloor \sqrt{I} \rfloor^2}{\lfloor \sqrt{I} \rfloor} \le \lfloor \sqrt{I} \rfloor + \frac{2 \cdot \lfloor \sqrt{I} \rfloor}{\lfloor \sqrt{I} \rfloor} = \lfloor \sqrt{I} \rfloor + 2.$$

- Provar que $(I \lfloor \sqrt{I} \rfloor^2) \le 2 \cdot \lfloor \sqrt{I} \rfloor$ (quadrados perfeitos).
 - Seja SR_x um quadrado perfeito e R_x sua raiz.
 - Para dois quadrados perfeitos consecutivos SR_i e SR_i → SR_i − SR_i = R_i + R_i.
 - Em outras palavras: $SR_i SR_i = 2 \cdot R_i + 1$.
 - Seja X um quadrado n\u00e3o-perfeito e seja GSR_X o maior quadrado perfeito menor que X. Seja R_X a raiz de GSR_X.

• $X - GSR_X < 2 \cdot R_X$.

- $\lfloor \sqrt{I} \rfloor + (2 \cdot \mathcal{F}) + (\mathcal{F} \cdot \frac{\mathcal{F}}{\lfloor \sqrt{I} \rfloor}) = \lfloor \sqrt{I} \rfloor + \frac{I \lfloor \sqrt{I} \rfloor^2}{\lfloor \sqrt{I} \rfloor}$.
- Como $\lfloor \sqrt{I} \rfloor^2$ é o maior quadrado perfeito menor que $I \to (I \lfloor \sqrt{I} \rfloor^2) \le 2 \cdot \lfloor \sqrt{I} \rfloor$ (prova adiada).

$$\bullet \to \lfloor \sqrt{I} \rfloor + \tfrac{I - \lfloor \sqrt{I} \rfloor^2}{\lfloor \sqrt{I} \rfloor} \le \lfloor \sqrt{I} \rfloor + \tfrac{2 \cdot \lfloor \sqrt{I} \rfloor}{\lfloor \sqrt{I} \rfloor} = \lfloor \sqrt{I} \rfloor + 2.$$

- Provar que $(I \lfloor \sqrt{I} \rfloor^2) \le 2 \cdot \lfloor \sqrt{I} \rfloor$ (quadrados perfeitos).
 - Seja SR_x um quadrado perfeito e R_x sua raiz.
 - Para dois quadrados perfeitos consecutivos SR_i e SR_i → SR_i − SR_i = R_i + R_i.
 - Em outras palavras: $SR_j SR_i = 2 \cdot R_i + 1$.
 - Seja X um quadrado não-perfeito e seja GSR_X o maior quadrado perfeito menor que X. Seja R_X a raiz de GSR_X.

Demonstração.

•
$$\lfloor \sqrt{I} \rfloor + (2 \cdot \mathcal{F}) + (\mathcal{F} \cdot \frac{\mathcal{F}}{\lfloor \sqrt{I} \rfloor}) = \lfloor \sqrt{I} \rfloor + \frac{I - \lfloor \sqrt{I} \rfloor^2}{\lfloor \sqrt{I} \rfloor}$$
.

• Como $\lfloor \sqrt{I} \rfloor^2$ é o maior quadrado perfeito menor que $I \to (I - \lfloor \sqrt{I} \rfloor^2) \le 2 \cdot \lfloor \sqrt{I} \rfloor$ (prova adiada).

$$\bullet \to \lfloor \sqrt{I} \rfloor + \tfrac{I - \lfloor \sqrt{I} \rfloor^2}{\lfloor \sqrt{I} \rfloor} \le \lfloor \sqrt{I} \rfloor + \tfrac{2 \cdot \lfloor \sqrt{I} \rfloor}{\lfloor \sqrt{I} \rfloor} = \lfloor \sqrt{I} \rfloor + 2.$$

- Provar que $(I \lfloor \sqrt{I} \rfloor^2) \le 2 \cdot \lfloor \sqrt{I} \rfloor$ (quadrados perfeitos).
 - Seja SR_x um quadrado perfeito e R_x sua raiz.
 - Para dois quadrados perfeitos consecutivos SR_i e SR_i → SR_i − SR_i = R_i + R_i.
 - Em outras palavras: $SR_j SR_i = 2 \cdot R_i + 1$.
 - Seja X um quadrado não-perfeito e seja GSR_X o maior quadrado perfeito menor que X. Seja R_X a raiz de GSR_X.

•
$$X - GSR_X < 2 \cdot R_X$$
.



Demonstração.

•
$$\lfloor \sqrt{I} \rfloor + (2 \cdot \mathcal{F}) + (\mathcal{F} \cdot \frac{\mathcal{F}}{\lfloor \sqrt{I} \rfloor}) = \lfloor \sqrt{I} \rfloor + \frac{I - \lfloor \sqrt{I} \rfloor^2}{\lfloor \sqrt{I} \rfloor}$$
.

• Como $\lfloor \sqrt{I} \rfloor^2$ é o maior quadrado perfeito menor que $I \to (I - \lfloor \sqrt{I} \rfloor^2) \le 2 \cdot \lfloor \sqrt{I} \rfloor$ (prova adiada).

$$\bullet \to \lfloor \sqrt{I} \rfloor + \frac{I - \lfloor \sqrt{I} \rfloor^2}{\lfloor \sqrt{I} \rfloor} \le \lfloor \sqrt{I} \rfloor + \frac{2 \cdot \lfloor \sqrt{I} \rfloor}{\lfloor \sqrt{I} \rfloor} = \lfloor \sqrt{I} \rfloor + 2.$$

- Provar que $(I \lfloor \sqrt{I} \rfloor^2) \le 2 \cdot \lfloor \sqrt{I} \rfloor$ (quadrados perfeitos).
 - Seja SR_x um quadrado perfeito e R_x sua raiz.
 - Para dois quadrados perfeitos consecutivos SR_i e SR_i → SR_i − SR_i = R_i + R_i.
 - Em outras palavras: $SR_j SR_i = 2 \cdot R_i + 1$.
 - Seja X um quadrado não-perfeito e seja GSR_X o maior quadrado perfeito menor que X. Seja R_X a raiz de GSR_X .

•
$$X - GSR_X \leq 2 \cdot R_X$$
.



Grafo gerado por SIM é uma arborescência



- No início, C contém apenas s.
- Ao final da primeira iteração, A_f será uma arborescência, pois
- De forma análoga, a partir da segunda iteração, para cada

Grafo gerado por SIM é uma arborescência

- No início, C contém apenas s.
- Ao final da primeira iteração, A_f será uma arborescência, pois apenas caminhos de menor custo a partir de s são adicionados
- De forma análoga, a partir da segunda iteração, para cada (x_{u,v}), apenas caminhos de custo mínimo entre u e t são adicionados a A_f.
 - Estes caminhos são formados por nós apenas de U (exceto para comprimeiro nó u) → inexistência de ciclos e múltiplos caminhos entre nós.

Grafo gerado por SIM é uma arborescência •••••

- No início, *C* contém apenas *s*.
- Ao final da primeira iteração, A_f será uma arborescência, pois apenas caminhos de menor custo a partir de s são adicionados.
- De forma análoga, a partir da segunda iteração, para cada $(x_{u,v})$, apenas caminhos de custo mínimo entre u e t são adicionados a A_t
 - Estes caminhos são formados por nós apenas de U (exceto para of primeiro nó u) → inexistência de ciclos e múltiplos caminhos entre nós.

Grafo gerado por SIM é uma arborescência

- No início, C contém apenas s.
- Ao final da primeira iteração, A_f será uma arborescência, pois apenas caminhos de menor custo a partir de s são adicionados.
- De forma análoga, a partir da segunda iteração, para cada (x_{u,v}), apenas caminhos de custo mínimo entre u e t são adicionados a A_f.
 - Estes caminhos s\(\tilde{a}\) formados por n\(\tilde{s}\) apenas de \(U\) (exceto para o primeiro n\(\tilde{o}\) u) → inexist\(\tilde{e}\) ncia de ciclos e m\(\tilde{u}\) ltiplos caminhos entre n\(\tilde{o}\)s.

Grafo gerado por SIM é uma arborescência •vota

- No início, C contém apenas s.
- Ao final da primeira iteração, A_f será uma arborescência, pois apenas caminhos de menor custo a partir de s são adicionados.
- De forma análoga, a partir da segunda iteração, para cada (x_{u,v}), apenas caminhos de custo mínimo entre u e t são adicionados a A_f.
 - Estes caminhos são formados por nós apenas de *U* (exceto para o primeiro nó *u*) → inexistência de ciclos e múltiplos caminhos entre nós.



- Seja t_h o terminal $u \in Next_{ter}^{\sqrt{I}}$ cujo valor de dist(s, u, G) é máximo.
- Todos os caminhos adicionados a A_f na iteração i terão custo

- Haverá no máximo $|\sqrt{I}| + 2$ iterations (Lema 9).
- Seja t um terminal adicionado a A_t na iteração j.
- t pode ser alcançado a partir de s através de uma série de
- Em cada iteração, os terminais em Next_{ter} seguem uma ordem
- O teorema segue.





- Seja t_h o terminal $u \in Next_{ter}^{\sqrt{I}}$ cujo valor de dist(s, u, G) é máximo.
- Todos os caminhos adicionados a A_f na iteração *i* terão custo $\leq k \cdot dist(s, t_h, G).$
 - $(x_{u,v},t)$ pertencerá a ε se
- Haverá no máximo $|\sqrt{I}| + 2$ iterations (Lema 9).
- Seja t um terminal adicionado a A_t na iteração j.
- t pode ser alcançado a partir de s através de uma série de
- Em cada iteração, os terminais em *Next*¹ seguem uma ordem
- O teorema segue.





- Seja t_h o terminal $u \in Next_{ter}^{\sqrt{I}}$ cujo valor de dist(s, u, G) é máximo.
- Todos os caminhos adicionados a A_f na iteração *i* terão custo $\leq k \cdot dist(s, t_h, G).$
 - $(x_{u,v},t)$ pertencerá a ε se $C(u, v) + dist(v, t, G(U)) \le k \cdot dist(s, t, G) \le k \cdot dist(s, t_h, G)$.
- Haverá no máximo $|\sqrt{I}| + 2$ iterations (Lema 9).
- Seja t um terminal adicionado a A_t na iteração j.
- t pode ser alcançado a partir de s através de uma série de
- Em cada iteração, os terminais em *Next*¹ seguem uma ordem
- O teorema segue.



- Seja t_h o terminal $u \in Next_{ter}^{\sqrt{I}}$ cujo valor de dist(s, u, G) é máximo.
- Todos os caminhos adicionados a A_f na iteração *i* terão custo $\leq k \cdot dist(s, t_h, G).$
 - $(x_{u,v},t)$ pertencerá a ε se $C(u, v) + dist(v, t, G(U)) < k \cdot dist(s, t, G) < k \cdot dist(s, t_h, G).$
- Haverá no máximo $|\sqrt{I}| + 2$ iterations (Lema 9).
- Seja t um terminal adicionado a A_t na iteração j.
- t pode ser alcançado a partir de s através de uma série de
- Em cada iteração, os terminais em *Next*¹ seguem uma ordem
- O teorema segue.



- Seja t_h o terminal $u \in Next_{ter}^{\sqrt{I}}$ cujo valor de dist(s, u, G) é máximo.
- Todos os caminhos adicionados a A_f na iteração *i* terão custo $\leq k \cdot dist(s, t_h, G).$
 - $(x_{u,v},t)$ pertencerá a ε se $C(u, v) + dist(v, t, G(U)) < k \cdot dist(s, t, G) < k \cdot dist(s, t_h, G).$
- Haverá no máximo $|\sqrt{I}| + 2$ iterations (Lema 9).
- Seja t um terminal adicionado a A_t na iteração j.
- t pode ser alcançado a partir de s através de uma série de
- Em cada iteração, os terminais em *Next*¹ seguem uma ordem
- O teorema segue.





- Seja t_h o terminal $u \in Next_{ter}^{\sqrt{I}}$ cujo valor de dist(s, u, G) é máximo.
- Todos os caminhos adicionados a A_f na iteração *i* terão custo $\leq k \cdot dist(s, t_h, G).$
 - $(x_{u,v},t)$ pertencerá a ε se $C(u, v) + dist(v, t, G(U)) < k \cdot dist(s, t, G) < k \cdot dist(s, t_h, G).$
- Haverá no máximo $|\sqrt{I}| + 2$ iterations (Lema 9).
- Seja t um terminal adicionado a A_t na iteração j.
- t pode ser alcançado a partir de s através de uma série de caminhos adicionados a A_f em j-1 iterações.
- Em cada iteração, os terminais em $Next_{ter}^{\sqrt{I}}$ seguem uma ordem
- O teorema segue.





- Seja t_h o terminal $u \in Next_{ter}^{\sqrt{I}}$ cujo valor de dist(s, u, G) é máximo.
- Todos os caminhos adicionados a A_f na iteração *i* terão custo $\leq k \cdot dist(s, t_h, G).$
 - $(x_{u,v},t)$ pertencerá a ε se $C(u, v) + dist(v, t, G(U)) < k \cdot dist(s, t, G) < k \cdot dist(s, t_h, G).$
- Haverá no máximo $|\sqrt{I}| + 2$ iterations (Lema 9).
- Seja t um terminal adicionado a A_t na iteração j.
- t pode ser alcançado a partir de s através de uma série de caminhos adicionados a A_f em j-1 iterações.
- Em cada iteração, os terminais em Next_{ter} seguem uma ordem não-decrescente de custo.





- Seja t_h o terminal $u \in Next_{ter}^{\sqrt{I}}$ cujo valor de dist(s, u, G) é máximo.
- Todos os caminhos adicionados a A_f na iteração *i* terão custo $\leq k \cdot dist(s, t_h, G).$
 - $(x_{u,v},t)$ pertencerá a ε se $C(u, v) + dist(v, t, G(U)) < k \cdot dist(s, t, G) < k \cdot dist(s, t_h, G).$
- Haverá no máximo $|\sqrt{I}| + 2$ iterations (Lema 9).
- Seja t um terminal adicionado a A_t na iteração j.
- t pode ser alcançado a partir de s através de uma série de caminhos adicionados a A_f em j-1 iterações.
- Em cada iteração, os terminais em Next_{ter} seguem uma ordem não-decrescente de custo.
- O teorema segue.



- Procedimento $CompPar \rightarrow (\sqrt{|T|} + 2)O(|V|^3)$:
 - $\sqrt{|T|}$ + 2: número de iterações (Lema 2).
 - $O(|V|^3)$: construção da SPT para os candidatos a \sqrt{I} -bad node.
- Procedimento CompGraphFirstPh: $O(|V|^2)$ (construção da SPT).
- Construco da instância do MSC:
 - Composição de V_1 : $O(|V|^2)$.
 - Composição de ε : $O(|V|^3)$ (executar o algoritmo de APSP).
- Algoritmo guloso de [Chekuri and Kumar 2004]:
 - Depende da execução do algoritmo para o MCG.
 - O algoritmo para o MCG depende da execução de oráculo
 → O(|V|²|T|) no nosso caso.
 - Complexidade do algoritmo para o MCG: $O(|V|^3|T|^2)$.
 - O algoritmo para o MSC aplica iterativamente o algorito para o MCG → O((log |T|)(|V|³|T|²).
- Loop entre as linhas 8 e 12:
 - Linha 10 se beneficia da execução do algoritmo do APSP.
 - Linha 9: |D| é proporcional a $|S_{(u,v)}|$ que é proporcional a O(|V|).
 - Linha 8: $|V_2| = |T|$.
 - Complexidade do loop: |T|O(|V|).
- Complexidade final: $O((\log |T|)(|V|^3|T|^2)$.

- Procedimento $CompPar \rightarrow (\sqrt{|T|} + 2)O(|V|^3)$:
 - $\sqrt{|T|}$ + 2: número de iterações (Lema 2).
 - $O(|V|^3)$: construção da SPT para os candidatos a \sqrt{I} -bad node.
- Procedimento *CompGraphFirstPh*: $O(|V|^2)$ (construção da SPT).
- Construço da instância do MSC:
 - Composição de V_1 : $O(|V|^2)$.
 - Composição de ε : $O(|V|^3)$ (executar o algoritmo de APSP).
- Algoritmo guloso de [Chekuri and Kumar 2004]:
 - Depende da execução do algoritmo para o MCG.
 - O algoritmo para o MCG depende da execução de oráculo
 → O(|V|²|T|) no nosso caso.
 - Complexidade do algoritmo para o MCG: $O(|V|^3|T|^2)$.
 - O algoritmo para o MSC aplica iterativamente o algorito para o MCG → O((log |T|)(|V|³|T|²).
- Loop entre as linhas 8 e 12:
 - Linha 10 se beneficia da execução do algoritmo do APSP.
 - Linha 9: |D| é proporcional a $|S_{(u,v)}|$ que é proporcional a O(|V|).
 - Linha 8: $|V_2| = |T|$.
 - Complexidade do loop: |T|O(|V|).
- Complexidade final: $O((\log |T|)(|V|^3|T|^2)$.

- Procedimento $CompPar \rightarrow (\sqrt{|T|} + 2)O(|V|^3)$:
 - $\sqrt{|T|}$ + 2: número de iterações (Lema 2).
 - $O(|V|^3)$: construção da SPT para os candidatos a \sqrt{I} -bad node.
- Procedimento $CompGraphFirstPh: O(|V|^2)$ (construção da SPT).
- Construço da instância do MSC:
 - Composição de V_1 : $O(|V|^2)$.
 - Composição de ε : $O(|V|^3)$ (executar o algoritmo de APSP).
- Algoritmo guloso de [Chekuri and Kumar 2004]:
 - Depende da execução do algoritmo para o MCG.
 - O algoritmo para o MCG depende da execução de oráculo
 → O(|V|²|T|) no nosso caso.
 - Complexidade do algoritmo para o MCG: $O(|V|^3|T|^2)$.
 - O algoritmo para o MSC aplica iterativamente o algorito para o MCG $\rightarrow O((\log |T|)(|V|^3|T|^2)$.
- Loop entre as linhas 8 e 12:
 - Linha 10 se beneficia da execução do algoritmo do APSP.
 - Linha 9: |D| é proporcional a $|S_{(u,v)}|$ que é proporcional a O(|V|).
 - Linha 8: $|V_2| = |T|$.
 - Complexidade do loop: |T|O(|V|).
- Complexidade final: $O((\log |T|)(|V|^3|T|^2)$.

- Procedimento $CompPar \rightarrow (\sqrt{|T|} + 2)O(|V|^3)$:
 - $\sqrt{|T|}$ + 2: número de iterações (Lema 2).
 - $O(|V|^3)$: construção da SPT para os candidatos a \sqrt{I} -bad node.
- Procedimento $CompGraphFirstPh: O(|V|^2)$ (construção da SPT).
- Construço da instância do MSC:
 - Composição de V_1 : $O(|V|^2)$.
 - Composição de ε : $O(|V|^3)$ (executar o algoritmo de APSP).
- Algoritmo guloso de [Chekuri and Kumar 2004]:
 - Depende da execução do algoritmo para o MCG.
 - O algoritmo para o MCG depende da execução de oráculo $\rightarrow O(|V|^2|T|)$ no nosso caso.
 - Complexidade do algoritmo para o MCG: $O(|V|^3|T|^2)$.
 - O algoritmo para o MSC aplica iterativamente o algorito para o MCG $\to O((\log |T|)(|V|^3|T|^2)$.
- Loop entre as linhas 8 e 12:
 - Linha 10 se beneficia da execução do algoritmo do APSP.
 - Linha 9: |D| é proporcional a $|S_{(u,v)}|$ que é proporcional a O(|V|).
 - Linha 8: $|V_2| = |T|$.
 - Complexidade do loop: |T|O(|V|).
- Complexidade final: $O((\log |T|)(|V|^3|T|^2)$.

- Procedimento $CompPar \rightarrow (\sqrt{|T|} + 2)O(|V|^3)$:
 - $\sqrt{|T|}$ + 2: número de iterações (Lema 2).
 - $O(|V|^3)$: construção da SPT para os candidatos a \sqrt{I} -bad node.
- Procedimento $CompGraphFirstPh: O(|V|^2)$ (construção da SPT).
- Construço da instância do MSC:
 - Composição de V_1 : $O(|V|^2)$.
 - Composição de ε : $O(|V|^3)$ (executar o algoritmo de APSP).
- Algoritmo guloso de [Chekuri and Kumar 2004]:
 - Depende da execução do algoritmo para o MCG.
 - O algoritmo para o MCG depende da execução de oráculo
 → O(|V|²|T|) no nosso caso.
 - Complexidade do algoritmo para o MCG: $O(|V|^3|T|^2)$.
 - O algoritmo para o MSC aplica iterativamente o algorito para o MCG → O((log |T|)(|V|³|T|²).
- Loop entre as linhas 8 e 12:
 - Linha 10 se beneficia da execução do algoritmo do APSP.
 - Linha 9: |D| é proporcional a $|S_{(u,v)}|$ que é proporcional a O(|V|).
 - Linha 8: $|V_2| = |T|$.
 - Complexidade do loop: |T|O(|V|).
- Complexidade final: $O((\log |T|)(|V|^3|T|^2)$.

- Procedimento $CompPar \rightarrow (\sqrt{|T|} + 2)O(|V|^3)$:
 - $\sqrt{|T|}$ + 2: número de iterações (Lema 2).
 - $O(|V|^3)$: construção da SPT para os candidatos a \sqrt{I} -bad node.
- Procedimento $CompGraphFirstPh: O(|V|^2)$ (construção da SPT).
- Construço da instância do MSC:
 - Composição de V_1 : $O(|V|^2)$.
 - Composição de ε : $O(|V|^3)$ (executar o algoritmo de APSP).
- Algoritmo guloso de [Chekuri and Kumar 2004]:
 - Depende da execução do algoritmo para o MCG.
 - O algoritmo para o MCG depende da execução de oráculo
 → O(|V|²|T|) no nosso caso.
 - Complexidade do algoritmo para o MCG: $O(|V|^3|T|^2)$.
 - O algoritmo para o MSC aplica iterativamente o algorito para o MCG $\rightarrow O((\log |T|)(|V|^3|T|^2)$.
- Loop entre as linhas 8 e 12:
 - Linha 10 se beneficia da execução do algoritmo do APSP.
 - Linha 9: |D| é proporcional a $|S_{(u,v)}|$ que é proporcional a O(|V|).
 - Linha 8: $|V_2| = |T|$.
 - Complexidade do loop: |T|O(|V|).
- Complexidade final: $O((\log |T|)(|V|^3|T|^2)$.

- Procedimento $CompPar \rightarrow (\sqrt{|T|} + 2)O(|V|^3)$:
 - $\sqrt{|T|}$ + 2: número de iterações (Lema 2).
 - $O(|V|^3)$: construção da SPT para os candidatos a \sqrt{I} -bad node.
- Procedimento $CompGraphFirstPh: O(|V|^2)$ (construção da SPT).
- Construço da instância do MSC:
 - Composição de V_1 : $O(|V|^2)$.
 - Composição de ε : $O(|V|^3)$ (executar o algoritmo de APSP).
- Algoritmo guloso de [Chekuri and Kumar 2004]:
 - Depende da execução do algoritmo para o MCG.
 - O algoritmo para o MCG depende da execução de oráculo
 → O(|V|²|T|) no nosso caso.
 - Complexidade do algoritmo para o MCG: $O(|V|^3|T|^2)$.
 - O algoritmo para o MSC aplica iterativamente o algorito para o MCG $\rightarrow O((\log |T|)(|V|^3|T|^2)$.
- Loop entre as linhas 8 e 12:
 - Linha 10 se beneficia da execução do algoritmo do APSP.
 - Linha 9: |D| é proporcional a $|S_{(u,v)}|$ que é proporcional a O(|V|).
 - Linha 8: $|V_2| = |T|$.
 - Complexidade do loop: |T|O(|V|).
- Complexidade final: $O((\log |T|)(|V|^3|T|^2)$.

- Complexidade semelhante em algumas partes. O tamanho de V_2 é $\sqrt{|T|}$ ao invés de O(|T|).
- Linha 4: $O(|V|^3)$ (semelhante ao *APPROX*).
- Loop (5-7):
 - |√|T||: múmero máximo de iterações
 - ullet $O(|T|\sqrt{|T|})$: tamanho máximo de *Marked* na última iteração
 - As outras instruções se beneficiam dos resultados da APSP e da SPT.
 - Gomplexidade do loop: O(|T|²)
- Algoritmo para o MSC (linha 8): $O((\log \sqrt{|T|})(|V|^3|T|))$ (novo tamanho de V_2).
- Loop (10-15):
 - √|T| + 2): número máximo de iterações.
 - O(V): tamanho de D
 - O(1) para as outras instruções (também se beneficial do resultado do APSP e da SPT).
- Complexidade final: $O((\log \sqrt{|T|})(|V|^3|T|\sqrt{|T|}))$.



- Complexidade semelhante em algumas partes. O tamanho de V_2 é $\sqrt{|T|}$ ao invés de O(|T|).
- Linha 4: $O(|V|^3)$ (semelhante ao *APPROX*).
- Loop (5-7):
 - |√|T||: múmero máximo de iterações
 - ullet $O(|T|\sqrt{|T|})$: tamanho máximo de *Marked* na última iteração
 - As outras instruções se beneficiam dos resultados da APSP e da SPT.
 - Complexidade do loop: O(|T|²)
- Algoritmo para o MSC (linha 8): $O((\log \sqrt{|T|})(|V|^3|T|))$ (novo tamanho de V_2).
- Loop (10-15):
 - $\sqrt{|T|} + 2$): número máximo de iterações
 - O(V): tamanho de E
 - O(1) para as outras instruções (também se beneficial do resultado do APSP e da SPT).
- Complexidade final: $O((\log \sqrt{|T|})(|V|^3|T|\sqrt{|T|}))$



- Complexidade semelhante em algumas partes. O tamanho de V_2 é $\sqrt{|T|}$ ao invés de O(|T|).
- Linha 4: $O(|V|^3)$ (semelhante ao *APPROX*).
- Loop (5-7):
 - $\lfloor \sqrt{|T|} \rfloor$: múmero máximo de iterações.
 - $O(|T|\sqrt{|T|})$: tamanho máximo de *Marked* na última iteração.
 - As outras instruções se beneficiam dos resultados da APSP e da SPT.
 - Complexidade do loop: $O(|T|^2)$.
- Algoritmo para o MSC (linha 8): $O((\log \sqrt{|T|})(|V|^3|T|))$ (novo tamanho de V_2).
- Loop (10-15):
 - $\sqrt{|T|}$ + 2): número máximo de iterações.
 - \bullet O(V): tamanho de E
 - O(1) para as outras instruções (também se beneficial do resultado do APSP e da SPT).
- Complexidade final: $O((\log \sqrt{|T|})(|V|^3|T|\sqrt{|T|}))$.



- Complexidade semelhante em algumas partes. O tamanho de V_2 é $\sqrt{|T|}$ ao invés de O(|T|).
- Linha 4: $O(|V|^3)$ (semelhante ao *APPROX*).
- Loop (5-7):
 - $\lfloor \sqrt{|T|} \rfloor$: múmero máximo de iterações.
 - $O(|T|\sqrt{|T|})$: tamanho máximo de *Marked* na última iteração.
 - As outras instruções se beneficiam dos resultados da APSP e da SPT.
 - Complexidade do loop: $O(|T|^2)$.
- Algoritmo para o MSC (linha 8): $O((\log \sqrt{|T|})(|V|^3|T|))$ (novo tamanho de V_2).
- Loop (10-15):
 - $\bullet \ \sqrt{|T|} + 2$): número máximo de iterações
 - \circ O(V): tamanho de E
 - O(1) para as outras instruções (também se beneficial do resultado do APSP e da SPT).
- Complexidade final: $O((\log \sqrt{|T|})(|V|^3|T|\sqrt{|T|}))$.



- Complexidade semelhante em algumas partes. O tamanho de V_2 é $\sqrt{|T|}$ ao invés de O(|T|).
- Linha 4: $O(|V|^3)$ (semelhante ao *APPROX*).
- Loop (5-7):
 - $\lfloor \sqrt{|T|} \rfloor$: múmero máximo de iterações.
 - $O(|T|\sqrt{|T|})$: tamanho máximo de *Marked* na última iteração.
 - As outras instruções se beneficiam dos resultados da APSP e da SPT.
 - Complexidade do loop: $O(|T|^2)$.
- Algoritmo para o MSC (linha 8): $O((\log \sqrt{|T|})(|V|^3|T|))$ (novo tamanho de V_2).
- Loop (10-15):
 - $ightarrow \sqrt{|T|+2}$: número máximo de iterações
 - O(V): tamanho de I
 - O(1) para as outras instruções (também se beneficial do resultado do APSP e da SPT).
- Complexidade final: $O((\log \sqrt{|T|})(|V|^3|T|\sqrt{|T|}))$.



- Complexidade semelhante em algumas partes. O tamanho de V_2 é $\sqrt{|T|}$ ao invés de O(|T|).
- Linha 4: $O(|V|^3)$ (semelhante ao *APPROX*).
- Loop (5-7):
 - $\lfloor \sqrt{|T|} \rfloor$: múmero máximo de iterações.
 - $O(|T|\sqrt{|T|})$: tamanho máximo de *Marked* na última iteração.
 - As outras instruções se beneficiam dos resultados da APSP e da SPT.
 - Complexidade do loop: $O(|T|^2)$.
- Algoritmo para o MSC (linha 8): $O((\log \sqrt{|T|})(|V|^3|T|))$ (novo tamanho de V_2).
- Loop (10-15):
 - $\sqrt{|T|} + 2$): número máximo de iterações
 - \bullet O(V): tamanho de I
 - O(1) para as outras instruções (também se beneficial do resultado do APSP e da SPT).
- Complexidade final: $O((\log \sqrt{|T|})(|V|^3|T|\sqrt{|T|}))$.



- Complexidade semelhante em algumas partes. O tamanho de V_2 é $\sqrt{|T|}$ ao invés de O(|T|).
- Linha 4: $O(|V|^3)$ (semelhante ao *APPROX*).
- Loop (5-7):
 - $\lfloor \sqrt{|T|} \rfloor$: múmero máximo de iterações.
 - $O(|T|\sqrt{|T|})$: tamanho máximo de *Marked* na última iteração.
 - As outras instruções se beneficiam dos resultados da APSP e da SPT.
 - Complexidade do loop: $O(|T|^2)$.
- Algoritmo para o MSC (linha 8): $O((\log \sqrt{|T|})(|V|^3|T|))$ (novo tamanho de V_2).
- Loop (10-15):
 - $\sqrt{|T|} + 2$): número máximo de iterações
 - O(V): tamanho de L
 - O(1) para as outras instruções (também se beneficial do resultado do APSP e da SPT).
- Complexidade final: $O((\log \sqrt{|T|})(|V|^3|T|\sqrt{|T|}))$.



- Complexidade semelhante em algumas partes. O tamanho de V_2 é $\sqrt{|T|}$ ao invés de O(|T|).
- Linha 4: $O(|V|^3)$ (semelhante ao *APPROX*).
- Loop (5-7):
 - $\lfloor \sqrt{|T|} \rfloor$: múmero máximo de iterações.
 - $O(|T|\sqrt{|T|})$: tamanho máximo de *Marked* na última iteração.
 - As outras instruções se beneficiam dos resultados da APSP e da SPT.
 - Complexidade do loop: $O(|T|^2)$.
- Algoritmo para o MSC (linha 8): $O((\log \sqrt{|T|})(|V|^3|T|))$ (novo tamanho de V_2).
- Loop (10-15):
 - √(T) + 2): número máximo de iterações
 - O(1) para as outras instruções (também se beneficial do resultado do APSP e da SPT).
- Complexidade final: $O((\log \sqrt{|T|})(|V|^3|T|\sqrt{|T|}))$.



- Complexidade semelhante em algumas partes. O tamanho de V_2 é $\sqrt{|T|}$ ao invés de O(|T|).
- Linha 4: $O(|V|^3)$ (semelhante ao *APPROX*).
- Loop (5-7):
 - $\lfloor \sqrt{|T|} \rfloor$: múmero máximo de iterações.
 - $O(|T|\sqrt{|T|})$: tamanho máximo de *Marked* na última iteração.
 - As outras instruções se beneficiam dos resultados da APSP e da SPT.
 - Complexidade do loop: $O(|T|^2)$.
- Algoritmo para o MSC (linha 8): $O((\log \sqrt{|T|})(|V|^3|T|))$ (novo tamanho de V_2).
- Loop (10-15):
 - $\sqrt{|T|} + 2$): número máximo de iterações.
 - O(V): tamanho de D.
 - O(1) para as outras instruções (também se beneficial do resultado do APSP e da SPT).
- Complexidade final: $O((\log \sqrt{|T|})(|V|^3|T|\sqrt{|T|}))$.



- Complexidade semelhante em algumas partes. O tamanho de V_2 é $\sqrt{|T|}$ ao invés de O(|T|).
- Linha 4: $O(|V|^3)$ (semelhante ao *APPROX*).
- Loop (5-7):
 - $\lfloor \sqrt{|T|} \rfloor$: múmero máximo de iterações.
 - $O(|T|\sqrt{|T|})$: tamanho máximo de *Marked* na última iteração.
 - As outras instruções se beneficiam dos resultados da APSP e da SPT.
 - Complexidade do loop: $O(|T|^2)$.
- Algoritmo para o MSC (linha 8): $O((\log \sqrt{|T|})(|V|^3|T|))$ (novo tamanho de V_2).
- Loop (10-15):
 - $\sqrt{|T|} + 2$): número máximo de iterações.
 - O(V): tamanho de D.
 - O(1) para as outras instruções (também se beneficial do resultado do APSP e da SPT).
- Complexidade final: $O((\log \sqrt{|T|})(|V|^3|T|\sqrt{|T|}))$.



- Complexidade semelhante em algumas partes. O tamanho de V_2 é $\sqrt{|T|}$ ao invés de O(|T|).
- Linha 4: $O(|V|^3)$ (semelhante ao *APPROX*).
- Loop (5-7):
 - $\lfloor \sqrt{|T|} \rfloor$: múmero máximo de iterações.
 - $O(|T|\sqrt{|T|})$: tamanho máximo de *Marked* na última iteração.
 - As outras instruções se beneficiam dos resultados da APSP e da SPT.
 - Complexidade do loop: $O(|T|^2)$.
- Algoritmo para o MSC (linha 8): $O((\log \sqrt{|T|})(|V|^3|T|))$ (novo tamanho de V_2).
- Loop (10-15):
 - $\sqrt{|T|} + 2$): número máximo de iterações.
 - O(V): tamanho de D.
 - O(1) para as outras instruções (também se beneficial do resultado do APSP e da SPT).
- Complexidade final: $O((\log \sqrt{|T|})(|V|^3|T|\sqrt{|T|}))$.



- Complexidade semelhante em algumas partes. O tamanho de V_2 é $\sqrt{|T|}$ ao invés de O(|T|).
- Linha 4: $O(|V|^3)$ (semelhante ao *APPROX*).
- Loop (5-7):
 - $\lfloor \sqrt{|T|} \rfloor$: múmero máximo de iterações.
 - $O(|T|\sqrt{|T|})$: tamanho máximo de *Marked* na última iteração.
 - As outras instruções se beneficiam dos resultados da APSP e da SPT.
 - Complexidade do loop: $O(|T|^2)$.
- Algoritmo para o MSC (linha 8): $O((\log \sqrt{|T|})(|V|^3|T|))$ (novo tamanho de V_2).
- Loop (10-15):
 - $\sqrt{|T|} + 2$): número máximo de iterações.
 - O(V): tamanho de D.
 - O(1) para as outras instruções (também se beneficial do resultado do APSP e da SPT).
- Complexidade final: $O((\log \sqrt{|T|})(|V|^3|T|\sqrt{|T|}))$.



- Complexidade semelhante em algumas partes. O tamanho de V_2 é $\sqrt{|T|}$ ao invés de O(|T|).
- Linha 4: $O(|V|^3)$ (semelhante ao *APPROX*).
- Loop (5-7):
 - $\lfloor \sqrt{|T|} \rfloor$: múmero máximo de iterações.
 - $O(|T|\sqrt{|T|})$: tamanho máximo de *Marked* na última iteração.
 - As outras instruções se beneficiam dos resultados da APSP e da SPT.
 - Complexidade do loop: $O(|T|^2)$.
- Algoritmo para o MSC (linha 8): $O((\log \sqrt{|T|})(|V|^3|T|))$ (novo tamanho de V_2).
- Loop (10-15):
 - $\sqrt{|T|} + 2$): número máximo de iterações.
 - O(V): tamanho de D.
 - O(1) para as outras instruções (também se beneficial do resultado do APSP e da SPT).
- Complexidade final: $O((\log \sqrt{|T|})(|V|^3|T|\sqrt{|T|}))$.





Călinescu, G., Chekuri, C., Pál, M., and Vondrák, J. (2011). Maximizing a monotone submodular function subject to a matroid constraint. SIAM J. Comput., 40(6):1740–1766.

Chekuri, C. and Kumar, A. (2004).

Maximum coverage problem with group budget constraints and applications.

In *APPROX-RANDOM*, pages 72–83. Springer-Verlag New York, Inc.

Elkin, M. and Kortsarz, G. (2006).

An approximation algorithm for the directed telephone multicast problem.

Algorithmica, 45:569–583.

Elkin, M. and Solomon, S. (2009).

Narrow-shallow-low-light trees with and without steiner points.

In *ESA*, pages 215–226.



Elkin, M. and Solomon, S. (2011). Narrow-shallow-low-light trees with and without steiner points. SIAM J. Discret. Math., 25(1):181–210.



Fisher, M. L., Nemhauser, G. L., and Wolsey, L. A. (1978). An analysis of approximations for maximizing submodular set functions-ii.

In Polyhedral Combinatorics, volume 8 of Mathematical *Programming Studies*, pages 73–87. Springer Berlin Heidelberg.



Fraigniaud, P. (2001).

Approximation algorithms for minimum-time broadcast under the vertex-disjoint paths mode.

In Proceedings of the 9th Annual European Symposium on Algorithms, ESA '01, pages 440-451, London, UK. Springer-Verlag.



Khandekar, R., Kortsarz, G., and Nutov, Z. (2011). Network-design with degree constraints. In Goldberg, L., Jansen, K., Ravi, R., and Rolim, J., editors, Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, volume 6845 of Lecture Notes in Computer Science, pages 289–301. Springer Berlin.

Nemhauser, G. L., Wolsey, L. A., and Fisher, M. L. (1978). An analysis of approximations for maximizing submodular set functions-i. Mathematical Programming, 14:265–294.

