ALGORITMOS DE APROXIMAÇÃO

Marcus Ritt

INF 05010 — Junho de 2019

Outline

O que é um algoritmo de aproximação

- Algoritmos exatos: garantia de otimalidade
- Algoritmos heurísticos: nenhuma garantia
- Algoritmos de aproximação: com garantia de qualidade

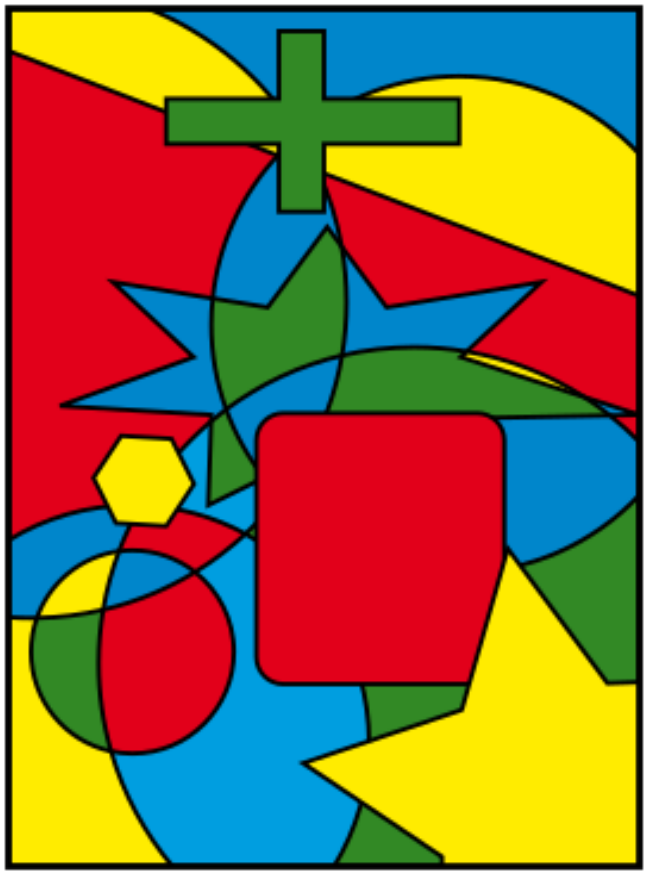
- Assume: solução ótima de valor 100, um algoritmo produz uma solução de valor 105.
- Logo: entre $(105 - 100)/100 = 5\%$ da otimalidade.
- Mais geral, para minimização, valor do algoritmo v, valor ótimo o temos desvio relativo

$$\frac{v - o}{o}$$

- Jargão: temos uma v/o-aproximação.
- Uma 1-aproximação dá a solução exata, uma 2-aproximação no máximo o dobro do mínimo.
- (Também existe a noção de aproximação absoluta, mas raramente se aplica.)

Excurso: teoremo de 4 cores

- Teorema: cada grafo planar é 4-colorível.
- Casos mais simples: 1-colorível? 2-colorível?
- Logo: questão 3 ou 4? $\rightarrow$ Aproximação absoluta!



Exemplo negativo: Problema do caixeiro viajante

Não existe uma α-aproximação em tempo polinomial para o PCV e para qualquer α, caso $P \neq NP$.

- Ou seja: nenhum algoritmo polinomial pode sempre retornar soluções de no máximo o dobro (ou um fator 3, 4, $\ldots$) da otimalidade caso $P \neq NP$.

Exemplo negativo: Problema do caixeiro viajante

- Porque acontece isso?
- Considera o problema do ciclo Hamiltoniano (CH) num grafo $G = (V, A)$.
- Vamos reduzir este problema para o PCV
  - Dá pesos de 1 para toda aresta $a \in A$.
  - Completa as demais arestas e atribui pesos αn a eles.
- Agora: caso existe um ciclo Hamiltoniano a solução ótima do PCV correspondente é n.
- Caso contrário: a solução ótima tem valor pelo menos $\alpha n + n - 1 > \alpha n$
- Logo: um algoritmo em tempo polinomial com garantia α para o PCV decide CH em tempo polinomial!
- (Esse estratégia se chama a técnica do intervalo (gap technique) e tem muitas aplicações.)

Exemplo positivo: Problema do caixeiro viajante métrico

- PCV de novo, mas agora o caso métrico: caminho direta entre duas cidades sempre é no máximo igual ao caminho que passa por outra cidade.
- Matematicamente: para cidades u, v, w

$$d_{uw} \leq d_{uv} + d_{vw} \qquad (\Delta)$$

- Cristofides (1976)

  Existe uma $3/2$-aproximação em tempo polinomial para o PCV.

- Ou seja: o valor da solução que o algoritmo de Cristofides retorna é nunca mais que 50% da otimalidade.
- Na prática frequentemente é melhor.

Como analisar um algoritmo de aproximação A?

- Como demonstrar que temos uma α-aproximação?
- Para cada instância x algoritmo A retorna uma solução $A(x)$ de valor $\varphi(A(x))$.
- Vamos supor que a instância possui um valor ótimo de $\mathsf{OPT}(x)$.
- Logo temos que demonstrar, que para toda instância possível x

$$\frac{\varphi(A(x))}{\mathsf{OPT}(x)} \leq \alpha \tag{1}$$

- (Maximização: similar.)
- Obstáculo: em geral não sabemos $\mathsf{OPT}(x)$!

- Como superar isso?
- Uma solução: focar num limitante inferior $\mathsf{LB}(x)$ para o valor da solução ótima $\mathsf{OPT}(x)$.
- Ou seja: $\mathsf{LB}(x) \leq \mathsf{OPT}(x)$.
- Com isso: caso

$$\frac{\varphi(A(x))}{\mathsf{LB}(x)} \leq \alpha$$

também temos

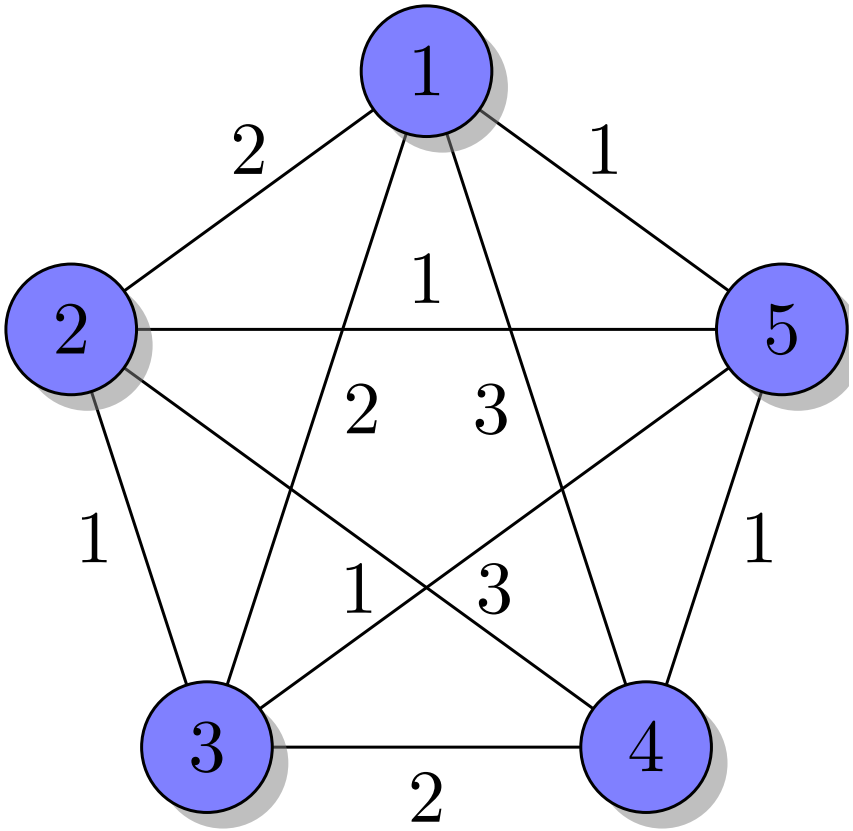
$$\frac{\varphi(A(x))}{\mathsf{OPT}(x)} \leq \alpha$$

Exemplo positivo: Problema do caixeiro viajante métrico

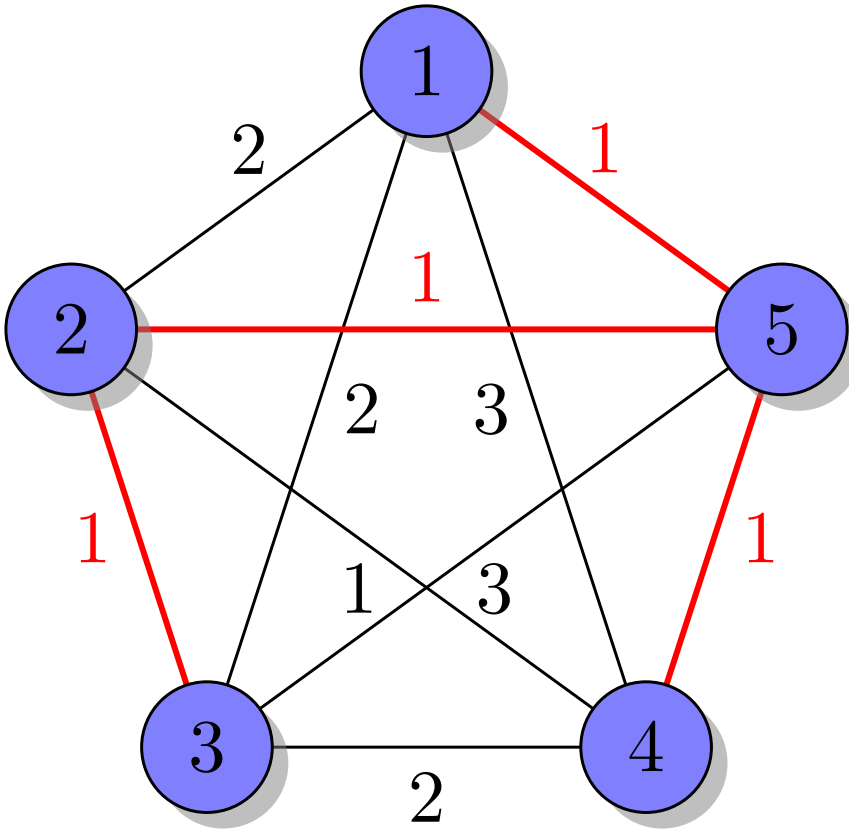
- Agora: vamos aplicar isso para o problema do caixeiro viajante métrico.
- Vamos analisar um algoritmo CHR_0 mais simples: dado $G = (V, A)$ com distâncias d
  1. Computa uma árvore geradora mínima T
  2. Duplica as arestas de T para obter T'
  3. Computa um caminho Euleriano C em T'
  4. Para obter uma rota R que visita cada vértice somente uma vez: remove cidades repetidas em C.
- Afirmação

  Algoritmo CHR_0 é uma 2-aproximação para o PCV. Ele roda em tempo $O(m \log n)$.

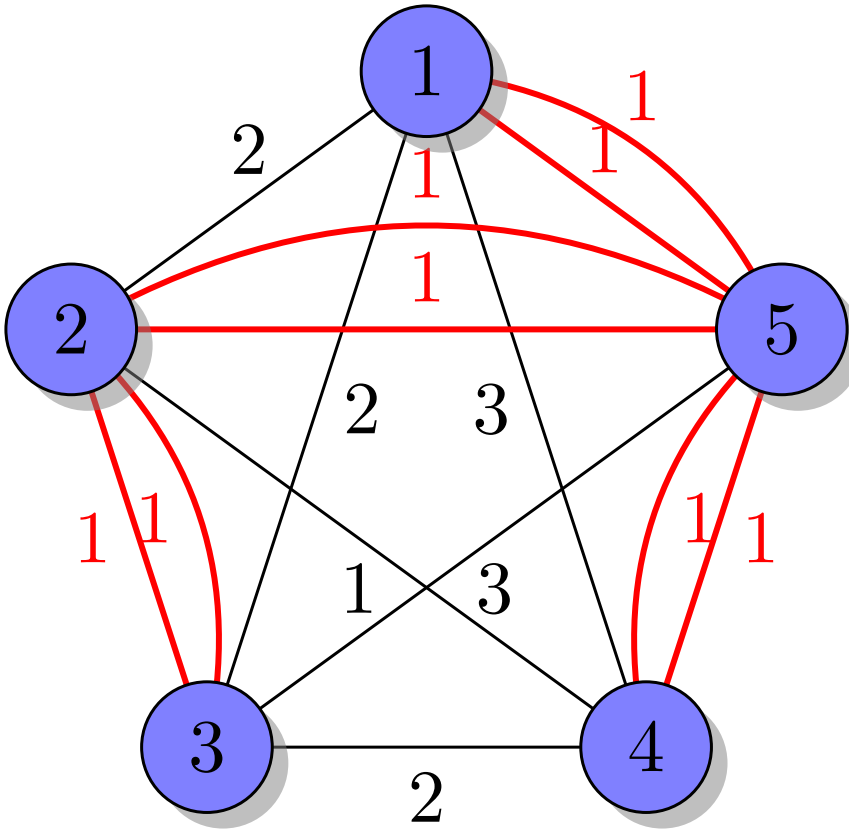
Exemplo: uma instância do PCV



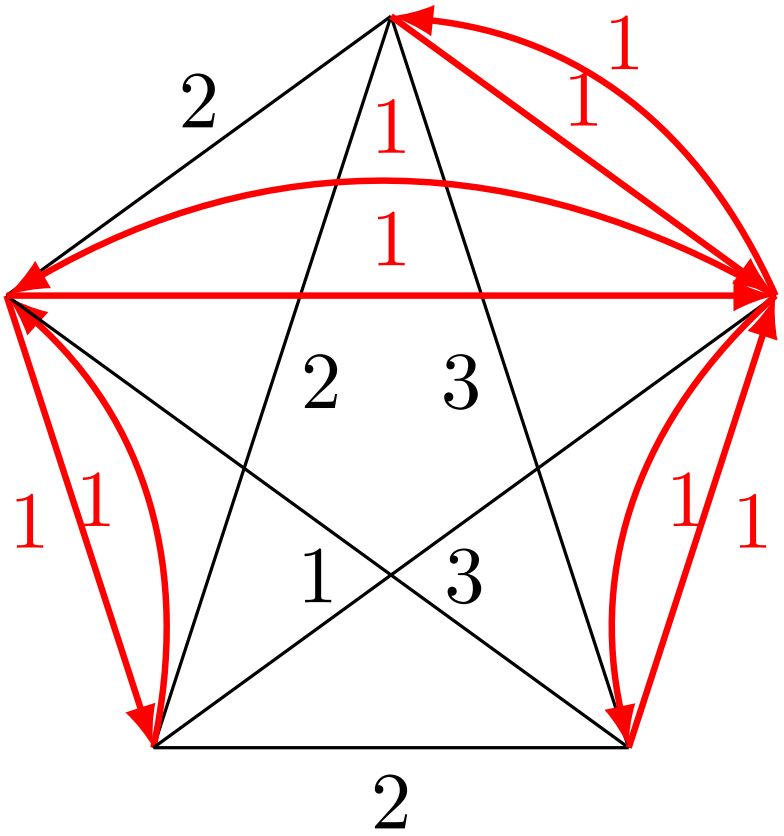
Exemplo: a árvore geradora mínima



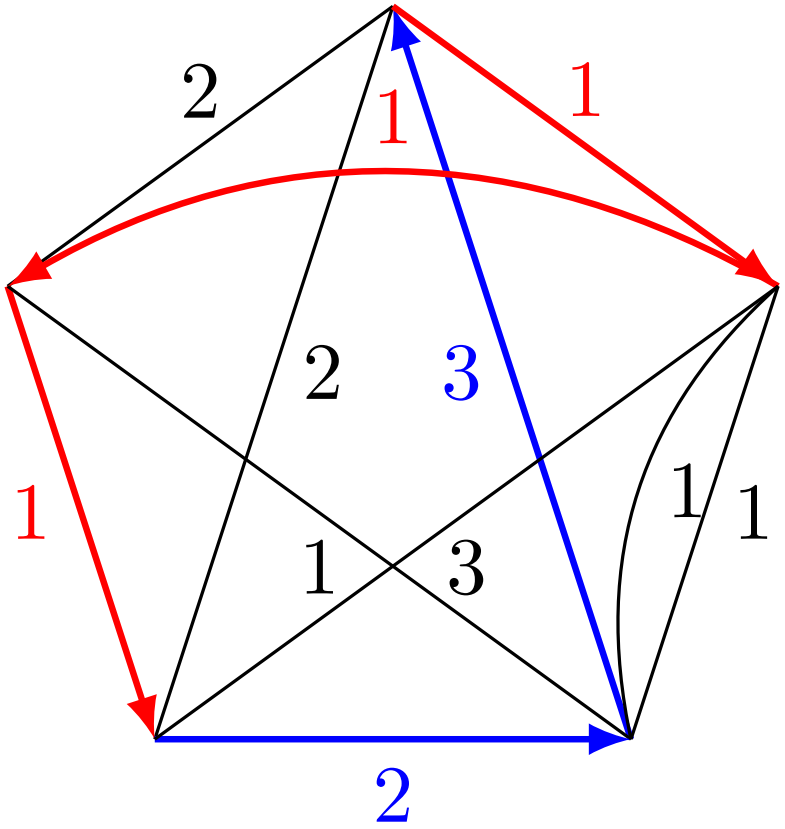
Exemplo: a árvore geradora mínima com arestas duplicadas



Exemplo: o caminho Euleriano



Exemplo: o rota depois de remover vértices repetidos



- Exercício: essa solução é ótima?

A complexidade é mais simples de analisar:

- Computar uma árvore geradora mínima: $O(m + n \log n)$ com o algoritmo de Prim.
- Duplicar as arestas: $O(n)$.
- Computar uma caminha Euleriano (Hierholzer): $O(m)$
- Remover cidades repetidas: $O(n)$.
- Total: $O(m + n \log n)$ dominado pelo árvore geradora mínima.

Análise, parte 2: corretude

- Para isso precisamos um limitante inferior.
- De fato: removendo uma rota obtemos uma árvore geradora.
- Logo: o custo de uma árvore geradora mínima é um limitante inferior para o custo da rota ótima

$$c(T) \leq \mathsf{OPT}(x)$$

- Mas: o custo $c(T')$ de árvore com arestas duplicada é $c(T') = 2c(T)$.
- Ainda: o custo do caminho Euleriano é exatamente $c(T')$.
- Ainda: remover arestas no último passo não produz uma rota mais longa que $c(T')$, pela desigualdade triangular (Δ).
- Tudo junto:

$$c(R) \leq c(T') = 2c(T) \leq 2\mathsf{OPT}(x)$$

- Tudo junto:

$$c(R) \leq c(T') = 2c(T) \leq 2\mathsf{OPT}(x)$$

- Logo: temos uma 2-aproximação!
- Notam como a desigualdade triangular é importante no último passo.
- Sem ela não temos essa garantia, porque pular cidades pode produzir rotas mais longas.

Agora:

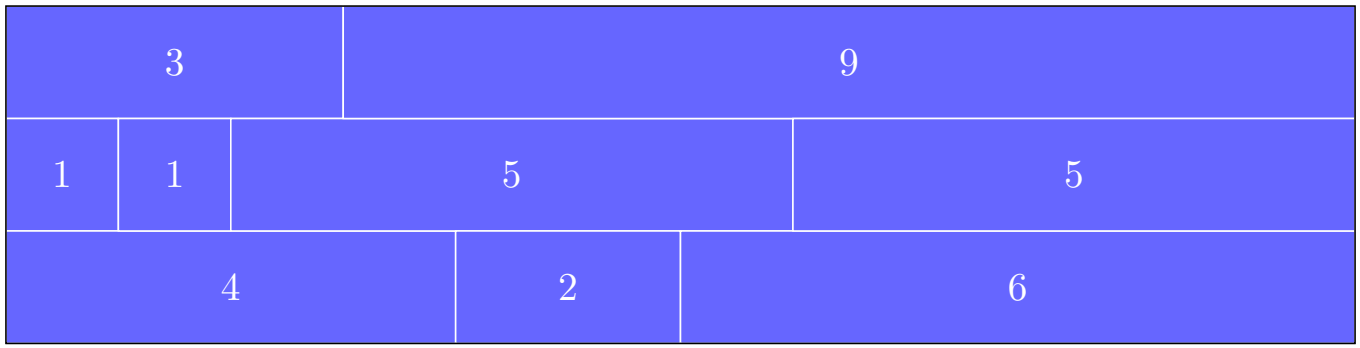
- Com um algoritmo melhor é possível obter uma $3/2$-aproximação.
- Isso foi o trabalho do Cristofides.

- Considere problema $P \,||\, C_{\max}$:
  - n processos tem que executar em m máquinas, sem interupção, sem sobreposição
  - Os processos tem tempos de processamente $p_1, \ldots, p_n$.
  - Qual o melhor agendamento que minimiza o makespan (i.e. o tempo de término da última tarefa)?

Exemplo: $n = 9$, $m = 3$

i	1	2	3	4	5	6	7	8	9
p_i	3	1	4	1	5	9	2	6	5

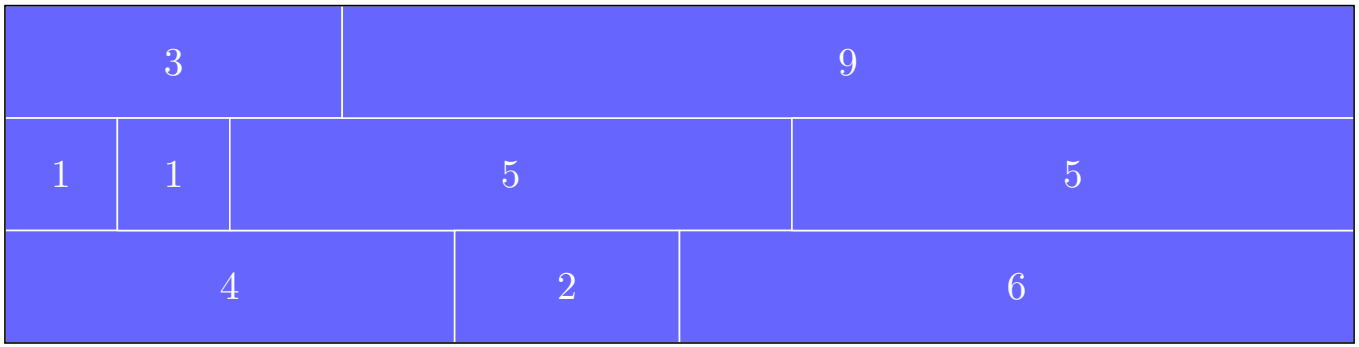
Colocando sempre na máquina de menor tempo de término atual:



- O algoritmo se chama list scheduling.
- Neste caso: solução ótima! Será que sempre é assim?

Exemplo: caso ruim para list scheduling

i	1	2	3	4	5	6	7
p_i	1	1	1	1	1	1	3



- Solução ótima: 3, solução encontrada: 5
- $\Longrightarrow$: list scheduling não pode ser melhor que uma $5/3$-aproximação
- É possível analisar melhor: a aproximação não pode ser melhor que $2 - 1/m$. Exercício!

Garantias para list scheduling

- É possível dar dois limitantes inferiores
  - O tempo média $\bar{p} = \sum_{i \in [n]} p_i / m$.
  - O tempo máximo $p_{\mathsf{max}} = \max_{i \in [n]} p_i$.
  - Logo: $\mathsf{LB}(x) = \max\{\bar{p}, p_{\mathsf{max}}\}$.
- Com isso podemos demonstrar:

  List scheduling é uma 2-aproximação.

List scheduling é uma 2-aproximação.

1. Em algum momento no processamento vamos colocar uma tarefa crítica j em alguma máquina tal que o tempo de término da tarefa define o makespan. Seja C o tempo atual da máquina selecionada, então $C_{\max} = C + p_j$.
2. Neste momento também temos

$$C \leq \sum_{1 \leq i < j} p_i/m \leq \bar{p}$$

   porque pela seleção da máquina de menor tempo C tem que ser menos que a média.
3. Além disso o tempo $p_j \leq p_{\max}$.
4. Juntando

$$C_{\max} = C + p_j \leq \bar{p} + p_{\max} \leq 2\mathsf{LB}(x).$$

Resumo

- Algoritmos de aproximação: fornecem uma garantia
- Em contrapartida a análise é mais difícil: temos que demonstrar ou
  - que uma aproximação não existe (exemplo: técnica do intervalo)
  - ou que a garantia sempre é satisfeito (exemplo: comparar a qualidade com um limitante inferior)
  - e ainda, no segunda: mostrar que a análise é a melhor possível

Para aprender mais sobre algoritmos de aproximação:

- Nova disciplina eletiva : Algoritmos de aproximação
- Professor : Rafael Santos Coelho

1. A solução do PCV no slide 16 é ótima?
2. Propõe uma melhor análise de list scheduling mostrando que a qualidade de aproximação não pode ser melhor que $2 - 1/m$.
3. Faça uma análise da complexidade de tempo do list scheduling.