Algoritmos

Pedro Hokama

Fontes

- [*clrs*] Algoritmos: Teoria e Prática (Terceira Edição) Thomas H. Cormen, Charles Eric Leiserson, Ronald Rivest, Ronald L. Rivest e Clifford Stein.
- [*timr*] Algorithms Illuminated Series, Tim Roughgarden

Apresentação Baseada:

- Stanford Algorithms
  `https://www.youtube.com/playlist?list=PLXFMmlk03Dt7Q0xr1PIAriY5623cKiH7V`
  `https://www.youtube.com/playlist?list=PLXFMmlk03Dt5EMI2s2WQBsLsZl7A5HEK6`
- Conjunto de Slides dos Professores Cid C. de Souza, Cândida N. da Silva, Orlando Lee, Pedro J. de Rezende
- Conjunto de Slides do Professores Cid C. de Souza para a disciplina MO420

Qualquer erro é de minha responsabilidade.

Paradigmas de Projeto de Algoritmos

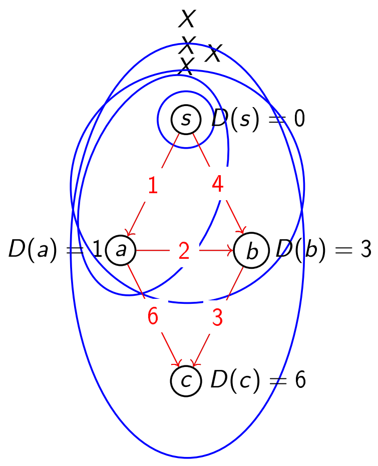
- Divisão e Conquista
- Aleatorização
- **Algoritmos Gulosos**
- Programação Dinâmica
- etc, etc...

Algoritmos Gulosos (Ganânciosos)

- Informalmente, um algoritmo guloso toma uma decisão que parece ser a melhor possível naquele momento (Escolha Gulosa).
- Não faz uma analise global para tomar essa decisão.
- Não se “arrepende” dessa decisão.
- Já vimos pelo menos um Algoritmo Guloso: O Algoritmo de Dijkstra.

Algoritmo de Dijkstra

- Manter um conjunto X com os vértices já processados.
- Considerar os arcos que começam em X e terminam em $V \setminus X$
- Escolher o arco (u, v) que minimiza $D(u) + c_{(u,v)}$, ou seja, o arco que minimiza o caminho para um vértice de $V \setminus X$.
- Note que não existe um caminho menor para chegar em v.
- Computa $D(v)$ e inclui v em X.
- Repita até que todos os vértices estejam em X, ou não tenha nenhum arco.



Algoritmos Gulosos - Vantagens

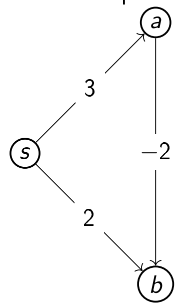
- Normalmente é fácil criar um critério guloso.
- Normalmente é fácil de implementar.
- Normalmente a complexidade é baixa e fácil de analisar.
- Desvantagens: A maioria dos critérios gulosos não vai levar a uma solução correta. pause Provar que um critério guloso leva a solução pode ser trabalhoso.

Algoritmos Gulosos - Contexto

- Estamos interessados em algoritmos que encontrem a solução correta para um dado problema.
- Algoritmos Gulosos também podem ser usados como heurísticas para encontrar boas soluções para problemas de otimização. Não necessariamente encontra a melhor solução, nem por isso estão incorretos.

Algoritmos Gulosos - Não Exemplo

- Suponha que temos o mesmo problema do caminho mínimo, mas agora com arestas de pesos negativos.



- Vamos tentar o mesmo critério guloso de Dijkstra.

Problema do Escalonamento Ponderado

- Suponha que você tenha uma máquina que faz alguma atividade.
- Trabalhos diferentes, cada trabalho i tem um tempo de execução l_i, e uma prioridade w_i, prioridade maior indica uma tarefa mais importante.
- Seja I o conjunto de todos os trabalhos, e c_i o tempo de conclusão do trabalho i.
- Em que ordem devo executar os trabalhos para minimizar

$$\sum_{i \in I} c_i \cdot w_i$$

Problema do Escalonamento Ponderado - Exemplo

- Três trabalhos $l_a = 1$, $l_b = 2$ e $l_c = 3$

0	1	2	3	4	5	6
a	b		c			

- Qual o tempo de término de cada atividade? $c_a = 1$, $c_b = 3$ e $c_c = 6$.
- Se as prioridades forem, $w_a = 3$, $w_b = 2$ e $w_c = 1$. Quanto é:

$$\sum_{i \in I} c_i \cdot w_i = 1 \cdot 3 + 3 \cdot 2 + 6 \cdot 1 = 15$$

Problema do Escalonamento Ponderado - Critério Guloso

- Vamos tentar criar alguns critérios gulosos.
- Caso 1: Todos os trabalhos tem comprimento igual, mas prioridades diferentes. Qual seria um bom critério guloso? Maior ou menor prioridade na frente?
- Caso 2: Todas as prioridades são iguais, mas o tempo de execução é diferente. Qual seria um bom critério guloso? Curtas ou Longas primeiro?
- Caso 3: Se você tiver um trabalho curto com alta prioridade, e um longo com baixa prioridade? Qual executar primeiro?
- Caso 4: Trabalho longo com alta prioridade e outro curto com baixa prioridade?

Problema do Escalonamento Ponderado - Critério Guloso

- Assim como no Dijkstra vamos tentar criar uma pontuação para guiar a nossa escolha.
- Essa pontuação tem que ser crescer se a prioridade for maior.
- E tem que diminuir quanto mais longa.
- Podemos pensar em pelo menos 2 tentativas claras:
- Tentativa 1: $w_i - l_i$
- Tentativa 2: $\frac{w_i}{l_i}$

Problema do Escalonamento Ponderado - Critério Guloso

a	$l_a = 5$	$w_a = 3$
b	$l_b = 6$	$w_b = 3$

- Tentativa 1: $w_i - l_i$
- Tentativa 2: $\frac{w_i}{l_i}$

Problema do Escalonamento Ponderado - Critério Guloso

- Assim como no Dijkstra vamos tentar criar uma pontuação para guiar a nossa escolha.
- Essa pontuação tem que ser crescer se a prioridade for maior.
- E tem que diminuir quanto mais longa.
- Podemos pensar em pelo menos 2 tentativas claras:
- Tentativa 1: $w_i - l_i$
- Tentativa 2: $\frac{w_i}{l_i}$

Problema do Escalonamento Ponderado - Critério Guloso

a	$l_a = 5$	$w_a = 3$
b	$l_b = 2$	$w_b = 1$

- Tentativa 1: $w_i - l_i$
- Tentativa 2: $\frac{w_i}{l_i}$

Tentativa 1:
$w_a - l_a = 3 - 5 = -2$
$w_b - l_b = 1 - 2 = -1$
Vai colocar b primeiro
$c_a = 7$ e $c_b = 2$
F.O. $= 7 \cdot 3 + 2 \cdot 1 = 23$

Tentativa 2:
$\frac{w_a}{l_a} = \frac{3}{5}$
$\frac{w_b}{l_b} = \frac{1}{2}$
Vai colocar a primeiro
$c_a = 5$ e $c_b = 7$
F.O. $= 5 \cdot 3 + 7 \cdot 1 = 22$

Problema do Escalonamento Ponderado - Corretude

- Suponha que renomeamos os trabalhos de forma que

$$\frac{w_1}{l_1} > \frac{w_2}{l_2} > \frac{w_3}{l_3} > \ldots > \frac{w_n}{l_n}$$

- O critério da Tentativa 2 irá escalonar os trabalhos nessa ordem $\sigma = 1, 2, 3, \ldots, n$.
- Suponha que exista uma outra sequencia $\sigma^* \neq \sigma$ tal que F.O.$(\sigma^*) <$ F.O.(σ)

Problema do Escalonamento Ponderado - Corretude

- Em σ^* vai existir duas atividades em sequencia, i, j tal que i é executado antes de j mas $i > j$.
  F.O. $(\sigma^*) = A + c_i \cdot w_i + c_j \cdot w_j + B$
- Suponha que trocamos essas duas atividades.

$$\begin{aligned}\text{novo custo} &= A + (c_i + l_j) \cdot w_i + (c_j - l_i) \cdot w_j + B \\ &= A + c_i \cdot w_i + l_j \cdot w_i + c_j \cdot w_j - l_i \cdot w_j + B\end{aligned}$$

- A diferença então é

$$l_j \cdot w_i - l_i \cdot w_j$$

Problema do Escalonamento Ponderado - Corretude

$$l_j \cdot w_i - l_i \cdot w_j$$

- Sabemos que se $i > j$, então

$$\begin{aligned}\frac{w_i}{l_i} &< \frac{w_j}{l_j} \\ \frac{(l_i \cdot l_j)w_i}{l_i} &< \frac{(l_i \cdot l_j)w_j}{l_j} \\ l_j \cdot w_i &< l_i \cdot w_j\end{aligned}$$

- portanto o que é economizado é maior que o gasto. Portanto a F.O. desse novo escalonamento é MENOR. Mas como σ^* era ótimo isso é um absurdo.

Problema do Escalonamento Ponderado - Complexidade

- Podemos calcular todas as razões em $O(n)$
- Ordenar em tempo $O(n \log n)$
- Portanto a complexidade total do algoritmo é $O(n \log n)$