Algoritmos gulosos

Monday, June 1, 2015 10:37 AM

27/06/15:

1: Interval scheduling

2: Seq. processos. $1 \| \sum C_i$

Hoje: 01/06/15

Algoritmos gulosos pt III.

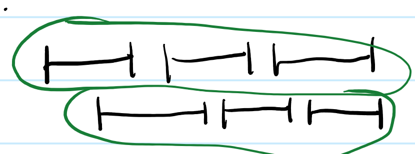
3. $1 \| \sum c_i w_i$

↳ ordenar por $w_i p_i$

4. <u>Interval scheduling #2</u>

Determinar a menor # de processadores para executar todas as tarefas.

Ex:

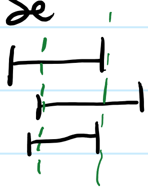


↳ 2 proc.

Alg. guloso:

a) Repetidamente aplica <u>I.S. 1</u>.

OBS: Se



Sobreposição $= \ell = 3$

então no mínimo $\ell = 3$ proc. são necessários.

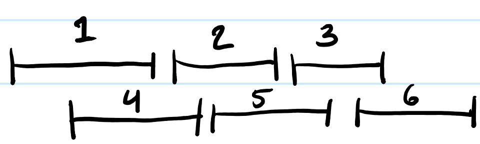
$\#proc \geq 3$ Obj: $\upsilon = \ell$.

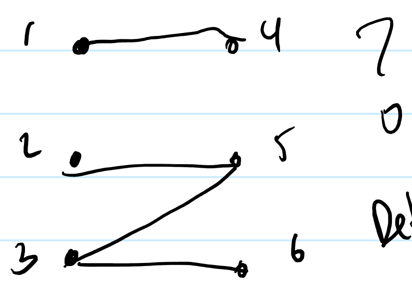
b) • Repetidamente processa os intervalos em ordem s_i não-decrescente

• Aloca cada intervalo no proc. livre

Acha $\upsilon \leq n$ processadores.

Interval scheduling 1: com grafos





Determinar conj. ind. max.

↳ Em geral, NP-completo

↳ Neste caso particular (grafos de intervalos) é NP.

Interval scheduling 2:

coloração de vértices.
Min color.

Árvores geradoras mínimas

Grafo ñ-direcionado e conexo $G=(V,A)$ e pesos $c_a \in \mathbb{Z}$, $a \in A$. Determinar o menor subgrafo conexo com arestas.

Variantes: Steiner Tree
↳ NP-completo

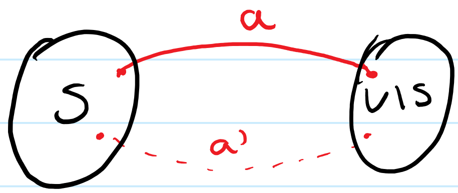
Solução gulosa:

- Ordenar arestas por peso
  $a_1 \leq a_2 \leq \dots \leq a_n$ com $w_1 \leq w_2 \dots \leq w_n$.
- Construir uma árvore tomando as arestas com menor custo que ~~não cobram vértices já cobertos~~ não gere ciclos.

Assumption: todos os pesos c_a são diferentes.

Prop: A aresta mais leve entre $S \subseteq V$, $S \neq 0$, e $V \setminus S$ faz parte da AGM.

prova: por contradição.
a) $c_{a'} > c_a$
b) conexo, $n-1 \rightarrow$ conexo.



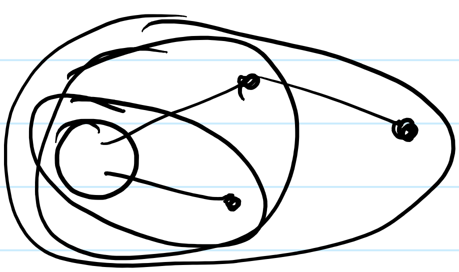
Mostrar que conectividade permanece trocando a por a' e soma dos pesos diminui.

OBS: S e V\S não são conexos (necessariamente).

Prop 2: Para um ciclo C, a aresta mais pesada em C ñ faz parte da AGM.

Alg. 2: Ordenar $c_{a_1} \geq c_{a_2} \geq c \ldots$
Remover arestas mantendo a conectividade.

Criar uma componente com um vértice S e deixar a componente crescer.



O(n) interações
Prim algorithm.
O(m · log n).
ou
Juntar componentes

Problema: manter componentes conexos. Estrutura de dados especial: union-find.

Kruskal: $O(m \cdot \log n)$.
Boruka. $O(m \cdot \log n)$
ou
Reverse-delete.
Problema: teste de conectiv.
$O(\log n (\log \log n)^3)$.