Segunda Prova - Otimização
Prof. André/Murilo
27 de junho de 2023

1. (10 pontos) Em uma resolução por *Branch & Bound* de um problema de maximização P, quais os critérios para que uma função B seja uma função limitante que possa ser usada nesta resolução? E o que é desejável que tenha para que seja uma "boa" função limitante?

2. (15 pontos) Considere o conjunto $S = \{1, 2, ..., n\}$ e seja $0 \leq k \leq n$. Proponha uma estratégia para enumerar (listar) todos os subconjuntos $A \subseteq S$ tal que $|A| = k$.

3. Considere uma árvore T, de raíz r, contendo n nós. Considere as funções $\text{PAI}(T, v)$ e $\text{FILHOS}(T, v)$ que retornam o nó pai e o conjunto de nós filhos do nó v na árvore T, respectivamente. Se r é a raíz, $\text{PAI}(T, r) = \lambda$. Um conjunto S de nós de T é dito um *conjunto independente* se para quaisquer $u, v \in S$, u não é pai de v e v não é pai de u. Queremos resolver o problema de encontrar um conjunto independente de tamanho máximo em T. Responda:

   (a) (15 pontos) Seja $I(v)$ o tamanho de um conjunto independente máximo da sub-árvore enraizada em v. Apresente uma descrição recursiva de $I(v)$?

   (b) (15 pontos) Considerando que o conjunto de nós de T são os números de 1 a n e que sempre que $u = \text{PAI}(v)$, temos que $u < v$, apresente um algoritmo de Programação Dinâmica que calcula $I(r)$.

4. Considere a seguinte recorrência que descreve as soluções de um problema R em função de subproblemas:

$$R(n, S) = \begin{cases} 0, & \text{se } n = 0, \\ \max\{R(n-1, S \setminus \{x\}) + v(x, n) \mid x \in S\}, & \text{caso contrário.} \end{cases}$$

   Onde, queremos resolver o problema $R(n, U)$, U é um conjunto tal que $|U| \geq n \geq 0$ elementos e $v : U \times \{1, \cdots, n\} \mapsto \mathbb{Z}$ é uma função.

   (a) (15 pontos) É possível usar a técnica de *Branch & Bound* para resolver R? Em caso afirmativo, descreva a solução. Como poderia ser uma função limitante?

   (b) (15 pontos) É possível usar a técnica de *Programação Dinâmica* para resolver R? Em caso afirmativo, descreva a solução. Neste caso, qual seria a estrutura de dados usada para guardar os subproblemas resolvidos e qual seria a ordem de execução?

5. (15 pontos) Considere o problema do *Caixeiro Viajante*, onde temos n cidades e queremos sair da cidade 1, passar por todas as outras cidades e voltar para a cidade 1 andando a menor distância possível. Considere que temos uma função de distâncias, $d(i, j)$, que nos diz a distância entre as cidades i e j. Considere também um algoritmo guloso para este problema que funciona assim: Iniciando da cidade 1, em cada passo escolhemos a cidade mais próxima. Este algoritmo funciona? Justifique.

Gabarito

1.

Seja X um subproblema (instância) de P e $OPT(X)$ o ótimo de P no subproblema X. Uma função limitante B para um problema de maximização deve ser tal que $B(X) \geq OPT(X)$, para todo X.

Uma função limitante "boa" é tal que $B(X)$ está "próximo" de $OPT(X)$ e tem custo baixo para ser calculada.

2.

Uma forma seria fazer usando um esquema de backtracking (com vetor característico ou sequência ordenada para representar os subconjuntos) e parar assim que o conjunto tiver extamente k elementos.

3.

(a) A ideia é a seguinte: ou a raíz da sub-árvore (v) não está no conjunto independente máximo ou está. No 1o caso, seus filhos podem estar ($\sum_{u \in \text{FILHOS}(T,v)} I(u)$). Já no segundo caso, os filhos não podem estar, mas os netos podem ($1 + \sum_{w \in \text{FILHOS}(T,u), \forall u \in \text{FILHOS}(T,v)} I(w)$).

$$I(v) = \max\{\sum_{u \in \text{FILHOS}(T,v)} I(u), 1 + \sum_{w \in \text{FILHOS}(T,u), \forall u \in \text{FILHOS}(T,v)} I(w)\}$$

(b) A estrutura de dados é um vetor, I, indexado de de 1 a n. O preenchimento deste vetor se dá do fim para o começo, já que sabemos que $\text{PAI}(v) < v$.

4.

(a) Sim. A ramificação é tal que em cada nó descrito por (n, S), temos um nó filho para cada $x \in S$. Uma função limitante $B(n, S)$ deve ser tal que $B(S, k) \geq R(n, S)$. Uma função limitante simples poderia ser $B(S, k) = \sum_{x \in S}(\max_{i \in [1..n]}\{v(x, i)\})$. Aqui, estamos escolhendo a posição (ordem) que maximiza $v(x, i)$ para cada $x \in S$, relaxando o fato que cada posição só pode ter um elemento de S.

(b) Sim. A indexação deve ser ajustada, já que um dos parâmetros é um subconjunto $S \subseteq U$. A estrutura de dados seria uma matriz $(n + 1) \times 2^{|U|}$. A base é quando $n = 0$ e a ordem de cálculo é de cima para baixo (n de 1 a n).

5.

Não funciona. Além do fato que o problema do caixeiro viajante (TSP) é um problema NP-difícil, temos facilmente contra-exemplos. Um contra-exemplo fácil tem 4 cidades com a seguinte matriz de distâncias:

	1	2	3	4
1	0	1	2	4
2	1	0	1	2
3	2	1	0	1
4	4	2	1	0

O algoritmo guloso dá como resposta a sequência 1-2-3-4, com custo 7, e uma resposta ótima seria a sequência 1-2-4-3, com custo 6.