Universidade Federal Fluminense
Curso: Ciência da Computação
Disciplina: Análise e Síntese de Algoritmos
Professor: Luís Felipe

Gabarito Lista de exercícios 5 – Algoritmos Randomizados e Aproximativos

1. Considere dois algoritmos aproximativos A e B para um mesmo problema de otimização. Digamos que a razão de aproximação do algoritmo A é $\frac{8}{5}$ e a razão de aproximação do algoritmo B é 2.

   (a) Se, para uma instância I, o algoritmo A retorna uma solução viável $A(I) = S$, satisfazendo $val(S) = 160$, quais os limitantes inferior e superior mais justos que podemos inferir para opt(I)?

   Resposta: Temos que $val(S) \leq \frac{8}{5} opt(I)$, e como $opt(I)$ é o mínimo, então $100 \leq opt(I) \leq 160$.

   (b) É possível que o algoritmo B retorne, para a mesma instância I do item a), uma solução viável $B(I) = T$, satisfazendo $val(T) = 80$? E uma solução viável X, satisfazendo $val(X) = 320$, é possível?

   Resposta: Nenhuma solução viável pode ter valor inferior a 100, portanto, não existe solução viável T satisfazendo $val(T) = 80$. Se $B(I) = X$, $val(X) = 320$ e $val(X) \leq 2opt(I)$, então $opt(I) \geq 160$. Obtemos de a) que $opt(I) = 160$, e assim, é possível $val(X) = 320$.

   (c) Sabendo agora que o algoritmo B retornou uma solução viável $B(I) = Y$, satisfazendo $val(Y) = 120$, para a mesma instância I do item a), para a qual obtivemos $A(I) = S$ satisfazendo $val(S) = 160$, o que podemos afirmar sobre opt(I)?

   Resposta: Se $B(I) = Y$, $val(Y) = 120$, então $60 \leq opt(I) \leq 120$, e pelo intervalo obtido do item a), temos $100 \leq opt(I) \leq 160$, logo $100 \leq opt(I) \leq 120$.

2. Considerando o problema $3SAT$ onde cada cláusula possui exatamente três literais, responda abaixo:

(a) Fixada uma clausula C qualquer, qual a probabilidade de satisfazermos C se escolhermos uma atribuição de suas variáveis aleatoriamente?

Resposta: Seja E uma expressão booleana com n variáveis $x_1, \cdots, x_n$ e m cláusulas, cada uma contendo três variáveis. Seja $C = (y_i \vee y_j \vee y_k)$ uma cláusula qualquer, em que $y_l \in \{x_l, \overline{x_l}\}$ para $l \in \{i, j, k\}$. Para que C não seja satisfeita por uma atribuição das variáveis $x_1, \cdots, x_n$, devemos ter $y_i = y_j = y_k = F$, enquanto as demais variáveis não relacionadas a C podem receber tanto F quanto V. Com isso, as variáveis associadas a C são fixas e as demais cada uma pode ter duas possibilidades de valores. Assim, o número de atribuições as variáveis que não satisfazem C é 2^{n-3}. Como há no total 2^n atribuições para todas as variáveis, pela complementaridade, temos que o número de atribuições que satisfazem C é $2^n - 2^{n-3} = 2^n(1 - \frac{1}{8}) = \frac{7}{8}2^n$. Portanto, a probabilidade desejada é $\frac{\frac{7}{8}2^n}{2^n} = \frac{7}{8}$.

(b) Mostre que cada instância de $3SAT$ existe uma atribuição de suas variáveis que satisfaz pelo menos $\frac{7}{8}$ das cláusulas.

Resposta: Seja E a expressão FNC com n variáveis e m cláusulas $C_1, \cdots, C_m$, cada uma com três literais. Seja X_i a variável indicadora tal que $X_i = 1$ se C_i é satisfeita, e $X_i = 0$ se C_i não é satisfeita. Como X_i é uma variável indicadora, temos que o valor esperado de X_i é precisamente sua probabilidade de sucesso, que pela questão a) vale $\frac{7}{8}$. Ou seja, $E(X_i) = \frac{7}{8}$. Assim, o número de cláusulas satisfeitas é $X = \sum_{i=1}^{k} X_i$. Pela linearidade da esperança, o valor esperado de X é $E(X) = E(\sum_{i=1}^{k} X_i) = \sum_{i=1}^{k} E(X_i) = \sum_{i=1}^{k} \frac{7}{8} = \frac{7}{8}m$. Como o valor esperado é $\frac{7}{8}m$ então há uma atribuição que satisfaz pelo menos $\frac{7}{8}m$, como segue da consequência do valor esperado.

3. Dado um grafo, considere a seguinte estratégia:
1) se existir vértice no grafo, tome um vértice de grau mínimo e adicione-o ao conjunto F, inicialmente vazio;
2) remova seus vizinhos e volte para o passo anterior.

(a) Ao final deste algoritmo, F retornará um conjunto de vértices que possui qual propriedade?

Resposta: Conjunto independente maximal.

(b) Para qual problema de otimização podemos aplicar este algoritmo? Este problema é de minimização ou maximização?

Resposta: Maximização.

(c) Exiba um exemplo de grafo para o qual o algoritmo retorne a solução ótima para o problema, **E** um outro exemplo para o qual o algoritmo não retorne a solução ótima para o problema.

Resposta: Considere a Figura 1. Na figura da esquerda o algoritmo retorna o conjunto independente máximo, enquanto o da direita, ao tomar o vértice a, a solução apresentada tem tamanho 2, enquanto que a solução ótima tem tamanho 3.

Um exemplo mais geral pode ser visto na Figura 2. Nele, o vértice x é vizinho a cada vértice v_i, $i = 1, \cdots, n$. Os vértices $v_1, \cdots, v_n$ formam um conjunto independente, e

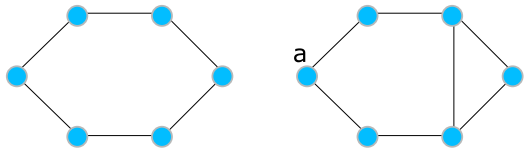

Figura 1: Figuras da Questão 3c. O exemplo da esquerda, a solução gulosa retorna a resposta ótima, enquanto o da direita, se escolhermos o vértice x na primeira etapa então a solução do algoritmo não será ótima.

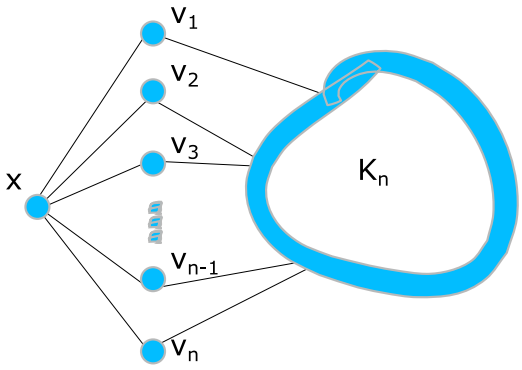

Figura 2: Figura da Questão 3c que mostra um caso que o algoritmo guloso nunca obtém a solução ótima e essa erro é em função linear do tamanho da entrada.

cada um desses vértices v_i é vizinho a todos os vértices do conjunto K_n. Esse último é um subgrafo completo com n vértices. O vértice x é o de menor grau no grafo, pois possui grau n, enquanto cada vértice v_i possui grau $n+1$ e os vértices de K_n possuem grau $n-1+n = 2n-1$. Dessa forma, a solução gulosa retornará resposta 2, com o vértice x e um vértice de K_n. Porém, a solução ótima tem tamanho n, com todos os vértices v_i, para $i = 1, \cdots, n$.

(d) Prove que a razão de aproximação deste algoritmo é $\frac{1}{\Delta+1}$, onde Δ é o grau máximo do grafo.

Resposta: Primeiramente, vamos argumentar que o algoritmo guloso retorna um conjunto independente S de tamanho $|S| \geq \frac{n}{\Delta+1}$, onde Δ é o grau máximo do grafo, ou seja, Δ o maior grau dentre todos os vértices do grafo.

Considere o conjunto $V \setminus S$, vamos apresentar um limite superior do número de vértices nesse conjunto da seguinte forma: Um vértice u pertence a $V \setminus S$ quando u é removido do grafo pelo fato dele ser vizinho a algum vértice $v \in S$ e v é adicionado a S pelo algoritmo guloso. Como cada vértice v em S possui no máximo Δ vizinhos, então $|V \setminus S| \leq \Delta|S|$. Ou seja, o número de vértices fora de S é limitado superiormente por cada vértice de S ter grau Δ. Como cada vértice exclusivamente pertence a S ou pertence a $V \setminus S$, temos que $|S| + |V \setminus S| = n$, e portanto: $|S| + \Delta|S| \geq n$, o que implica $|S| \geq \frac{n}{\Delta+1}$.

Como um limite superior para o tamanho de S é n, temos então a razão $\frac{\frac{n}{\Delta+1}}{n} = \frac{1}{\Delta+1}$.

4. Certo exame de sangue pode ser entendido como um algoritmo de Monte Carlo

baseado-no-sim que, com probabilidade $p \geq 95\%$, diagnostica determinada doença X caso o dono do sangue examinado de fato a possua. Uma epidemia de X fez com que um terço dos habitantes de uma cidade estivesse com aquela doença, cujo tratamento é, no entanto, muito penoso e não deve ser administrado a pessoas sãs. Quantas vezes aquele exame precisará ser repetido até possa ser avaliada como "desprezível" (menor do que 1%) a chance de que uma pessoa daquela cidade apresente a doença X?

Resposta: Considere os seguintes eventos associados a uma execução do algoritmo que consiste na aplicação do exame de sangue em um paciente:

- A_S: o algoritmo responde SIM
- A_N: o algoritmo responde NÃO
- C_S: a resposta correta para aquela entrada é SIM
- C_N: a resposta correta para aquela entrada é NÃO

Pelos dados do enunciado, temos que:

- $P(A_S|C_S) = 0,95$: Se a pessoa realmente está doente, o exame responde SIM com 95% de certeza.
- $P(A_N|C_S) = 0,05$: Existe 5% de chance de erro quando a pessoa está doente (ou seja, o exame não indica doença).
- $P(A_S|C_N) = 0$: Como o algoritmo (exame) é baseado no SIM, não há erro quando ele indica doença.
- $P(A_N|C_N) = 1$: Como o algoritmo (exame) é baseado no SIM, não há possibilidade de erro quando o paciente não está doente (ou seja, não há falso positivo).

Observe ainda, que nos são informados os valores $P(C_S) = 1/3$ e $P(C_N) = 2/3$.

Suponha que nosso algoritmo consiste não apenas em aplicar o exame uma vez, mas em repeti-lo k vezes. Neste caso, $P(A_N|C_S) = 0,05^k$ e $P(A_S|C_S) = 1 - 0,05^k$.

Nosso objetivo é descobrir, dado que o exame não indicou doença, qual a real probabilidade de o paciente estar doente (queremos que este valor seja menor que 1%). Ou seja, queremos saber $P(C_S|A_N)$. Temos que:

$$P(C_S|A_N) = \frac{P(C_S, A_N)}{P(A_N)} = \frac{P(A_N|C_S)P(C_S)}{P(A_N|C_S)P(C_S) + P(A_N|C_N)P(C_N)}$$

Substituindo pelos valores conhecidos,

$$P(C_S|A_N) = \frac{(0,05)^k(1/3)}{(0,05)^k(1/3) + 2/3}$$

Basta agora descobrir para que valores de k a expressão acima é menor do que 1%. Efetuando as contas, vemos que, para uma única repetição do exame, a chance de "falso negativo" é de $2,45\%$. Se efetuarmos dois exames, essa chance já será de $0,12\%$, atingindo nosso objetivo com apenas duas repetições.

5. Considere o problema Mínimo Steiner Tree, a entrada consiste de: Um grafo $G = (V, E)$ com um subconjunto de vértices terminais $V' \subseteq V$ e as distâncias entre todos os pares de vértices que satisfaçam a desigualdade triangular. O objetivo é determinar uma árvore com custo mínimo que possua todos os vértices de V'. Observe que essa árvore pode ou não ter vértices de $V \setminus V'$. Elabore um algoritmo 2-aproximativo para resolver o problema Mínimo Steiner Tree e determine sua complexidade.

   Resposta: Dado o grafo G, defina um grafo completo G' da seguinte forma: Cada vértice terminal do conjunto V' de G é vértice de G' e o peso de cada aresta é igual a distância dos dois vértices associados em G. O algoritmo faz o seguinte procedimento:

   1) Obtém uma MST em G'.
   2) De posse da árvore, recupere os caminhos associados no grafo G.

   A construção do novo grafo G' possui complexidade $O(|V'|)$, dado que temos de entrada o grafo G com as distâncias de todos os pares de vértices. A obtenção da MST em G' possui complexidade $O(|V'|^2 \log |V'|)$, dado que G' é um grafo completo. A recuperação dos caminhos associados em G consome tempo linear no grafo G para o caminho de cada par de vértices de G'. Temos assim, nessa etapa, a complexidade de $O(|V'|^2|V|)$. Dessa forma, a complexidade do algoritmo é $O(|V'|^2 \log |V'|) + O(|V'|^2|V|)$. Como $V' \subseteq V$, então $O(|V'|^2 \log |V'|) + O(|V'|^2|V|) = O(|V'|^2|V|)$.

   Vamos provar agora que esse algoritmo é uma 2-aproximação da solução ótima do problema Mínimo Steiner Tree.

   Considere T' a solução do algoritmo e T a solução ótima do Mínimo Steiner Tree de G. Note que não conhecemos a árvore T, mas vamos provar que $c(T') \leq 2c(T)$.

   Considere a árvore T, duplique a árvore T duplicando as arestas de T, obtendo um multigrafo. Note que o novo grafo é Euleriano. Mapeie os caminhos de terminais a terminais às arestas do circuito euleriano do novo grafo. Seja CE o circuito euleriano obtido.

   Assim, a soma dos tamanhos dos caminhos obtidos do circuito euleriano é igual ao dobro do custo de T, ou seja $c(|CE|) = 2c(T)$. Ao removermos arestas de ciclos associados circuito CE ainda garantindo a conexidade, temos uma árvore T'', portanto $c(T'') \leq 2c(T)$. Como T'' é uma árvore, então $c(T') \leq c(T'')$, pois T' é a árvore de custo mínimo, o que implica $c(T') \leq 2c(T)$.