Universidade Federal do Ceará – Departamento de Computação

CK0019–T02A–2014.2 Avaliação Parcial 2 – Parte 1 2014-10-17

1. (2,5 pontos) Escreva um algoritmo que receba uma entrada $(p[1..n], v[1..n], P, m)$ para o problema da mochila 0-1, onde $p[i]$ é o peso e $v[i]$ o valor do i-ésimo objeto, P é a capacidade máxima da mochila e m é o MDC dos inteiros P, $p[1]$, $p[2]$, ..., $p[n]$. O seu algoritmo deve resolver o problema para a entrada em questão em tempo $O(n * P/m)$, retornando como saída um vetor $e[1..n]$ de 0's e 1's que represente uma solução ótima, cada $e[i]$ valendo 1 se e somente se o i-ésimo objeto é escolhido na solução.

Resposta.

```
================================================================================
Algoritmo: mochila
Entrada: vetores p[1..n] e v[1..n] e números "P" e "m", como descrito acima.
Saída: vetor e[1..n] de 0's e 1's.
--------------------------------------------------------------------------------
01. PARA i DE 1 A n
02. | V[0,i] := 0 , E[0,i] := 0
03. V[0,0] := 0 , peso := m
04. ENQUANTO peso <= P
05. | V[peso,0] := 0
06. | PARA i DE 1 A n
07. | | V[peso,i] := V[peso,i-1] , E[peso,i] := 0
08. | | SE p[i] <= peso
09. | | | VS := v[i] + V[peso - p[i] , i-1]
10. | | | SE VS > V[peso,i]
11. | | | | V[peso,i] := VS , E[peso,i] := 1
12. | peso := peso + m
13. peso := P
14. PARA i DE n A 1 PASSO -1
15. | e[i] := E[peso,i]
16. | SE e[i] = 1
17. | | peso := peso - p[i]
18. RETORNE e.
================================================================================
```

O algoritmo acima preenche uma matriz $(P+1) \times (n+1)$, mas apenas as linhas de número múltiplo de m, e por isso executa em tempo $O(n * P/m)$, CQD. □

2. (2,5 pontos) Dado um vetor A de elementos $A[1]$, $A[2]$, ..., $A[n]$, com $n \geq 1$, seja $A[i..j]$ o subvetor de A que possui os elementos $A[i]$, $A[i+1]$, ..., $A[j]$, e seja $soma(A[i..j]) = \sum_{k=i}^{j} A[k]$. Seja ainda a *soma máxima* de A o máximo entre zero e a maior soma de um subvetor de A, ou seja, $\max\{0, \max\{soma(A[i..j]) : 1 \leq i \leq j \leq n\}\}$. Assim, por exemplo, a soma máxima de $A = [\ 1,\ 3,\ -3,\ 1,\ -3,\ 1,\ 1,\ -1,\ 4,\ -2,\ 1\ ]$ é 5, pois $soma(A[6..9]) = 5 \geq 0$ e não há subvetor de A de soma maior que 5.

Escreva e explique, então, um algoritmo que receba como entrada um vetor A de n números reais e, em tempo $O(n)$ e usando $O(1)$ de memória auxiliar, retorne a soma máxima de A.

Dica: calcule a soma máxima de $A[1..k+1]$ por meio da soma máxima de $A[1..k]$.

Resposta.

```
================================================================================
Algoritmo: Kadane
Entrada: vetor A[1..n] de números reais.
Saída: um número real.
--------------------------------------------------------------------------------
1. sm := 0 , si := 0
2. PARA i DE 1 A n
3. | si := máx{0, si} + A[i]
4. | sm := máx{sm, si}
5. RETORNE sm.
================================================================================
```

O algoritmo de Kadane calcula, para cada i de 1 a n, a maior soma de um subvetor que termine em $A[i]$ – armazenada em `si` – e a soma máxima de $A[1..i]$ – armazenada em `sm`. A linha 3 corresponde à observação de que a maior soma de um subvetor que termine em $A[i]$ é igual a $A[i]$ mais a maior soma de um subvetor que termine em $A[i-1]$, exceto se este último valor for negativo ou se $i = 1$; nestes casos, a soma em questão vale apenas $A[i]$, como calculado pelo algoritmo. A linha 4 corresponde à observação de que a soma máxima de $A[1..i]$ ou é a soma de um subvetor terminando em $A[i]$, ou é a soma máxima de $A[1..i-1]$ (se $i > 1$) ou é zero.

O algoritmo de Kadane executa em tempo $\Theta(n)$ e usa $O(1)$ de memória auxiliar, e portanto resolve o problema da soma máxima nas melhores complexidades de tempo e espaço possíveis. Ele é uma bela aplicação da técnica de programação dinâmica, e suas aplicações são estudadas até os dias de hoje. □

Universidade Federal do Ceará – Departamento de Computação

CK0019–T02A–2014.2 Avaliação Parcial 2 – Parte 2 2014-10-20

3. (1,5 pontos) Se T é um texto cujos caracteres ocorrem com estas frequências

caractere	a	b	c	d	e	f	g	h	i
ocorrências	1	3	7	5	2	11	8	4	6 ,

informe a árvore binária produzida pelo algoritmo de Huffman para T (mostre os seus cálculos).

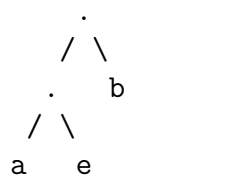
Resposta. O algoritmo começa pelos dois caracteres menos frequentes, a e e:

```
  .
 / \
a   e
```

Trocando a e e por um caractere ae, temos:

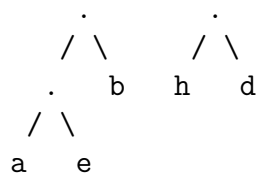
caractere	ae	b	c	d	f	g	h	i
ocorrências	3	3	7	5	11	8	4	6 .

Os caracteres menos frequentes são então ae e b, com o quê obtemos:



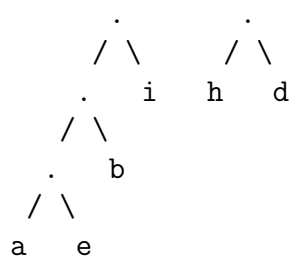
caractere	aeb	c	d	f	g	h	i
ocorrências	6	7	5	11	8	4	6 .

Os caracteres menos frequentes são agora h e d:



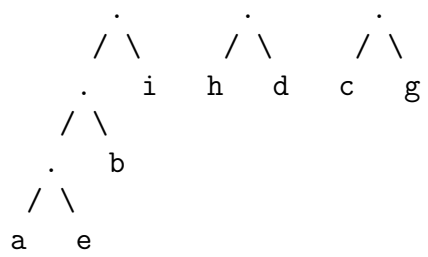
caractere	aeb	c	f	g	hd	i
ocorrências	6	7	11	8	9	6 .

Próxima escolha: i e aeb:



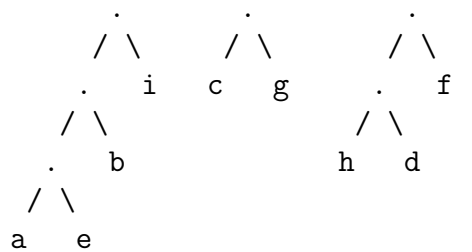
caractere	aebi	c	f	g	hd
ocorrências	12	7	11	8	9 .

Próxima escolha: c e g:



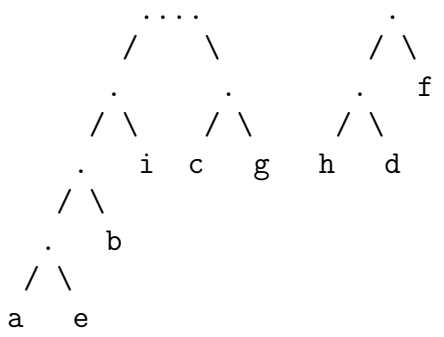
caractere	aebi	cg	f	hd
ocorrências	12	15	11	9 .

Próxima escolha: hd e f:



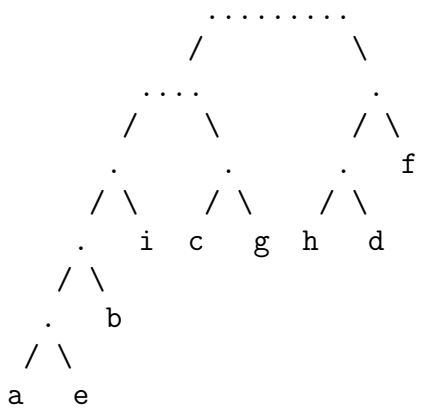
caractere	aebi	cg	hdf
ocorrências	12	15	20 .

Próxima escolha: *aebi* e *cg*:



caractere	aebicg	hdf
ocorrências	27	20 .

A última escolha é a única possível:



□

4. (2 pontos) Um comerciante revendia, na cidade onde morava, doces que ele comprava no interior. Em certa ida ao interior, ele encontrou n tipos de doce: do i-ésimo tipo havia $q[i]$ quilos à venda, custando $c[i]$ reais/quilo, e, pelas estimativas do comerciante, cada quilo poderia ser vendido por $v[i]$ reais na cidade, sendo $v[i] > c[i]$. O comerciante poderia comprar quantas gramas quisesse de cada doce, mas sabia que, acima de C quilos de carga total, seria multado em m reais por quilo de carga excessiva pela fiscalização intermunicipal. O veículo do comerciante tinha, porém, amplo espaço para a mercadoria, de forma que ele podia escolher pagar a multa por carga excessiva, se isso fosse lucrativo.

Assim sendo, escreva um algoritmo que resolva o *problema do vendedor de doces*, que é o de fazer a compra de doces que daria ao comerciante o maior lucro estimado. Todos os valores da entrada são reais positivos, e a saída deve ser um vetor $s[1..n]$, cada $s[i]$ sendo a quantidade a ser comprada do doce i (portanto, um número real de 0 a $q[i]$).

Resposta.

```
================================================================================
Algoritmo: vend_doces
Entrada: vetores q[1..n], c[1..n], v[1..n] e números "C" e "m"
Saída: vetor s[1..n]
```

```
--------------------------------------------------------------------------------
01. PARA i DE 1 A n
02. | l[i] := v[i] - c[i] , s[i] := 0
03. Crie um vetor I[1..n] que contenha os índices {1 ... n} numa ordem tal que
    l[ I[1] ] >= l[ I[2] ] >= ... >= l[ I[n] ].
04. i := 1
05. ENQUANTO i <= n E C > 0
06. | d := I[i] , qini := q[d] , qtd := mín{ C , qini } , s[d] := qtd ,
    | q[d] := q[d] - qtd , C := C - qtd
07. | SE qtd = qini
08. | | ++i
09. ENQUANTO i <= n
10. | d := I[i]
11. | SE l[d] <= m
12. | | RETORNE s.
13. | s[d] := s[d] + q[d] , ++i
14. RETORNE s.
================================================================================
```

O algoritmo acima executa em tempo $O(n \log n)$, que corresponde à ordenação da linha 3 (o restante do código executa em tempo $O(n)$).

Outra implementação possível utiliza um monte ("heap") para selecionar sucessivamente os doces de maior lucro. Tal implementação executa em tempo $O(n + k \log n)$, onde k é o número de doces efetivamente retirados do monte – logo, $k \leq n$. Essa implementação tem potencial para executar mais rapidamente que a anterior nos casos em que k é consideravelmente menor que n. □

5. (1,5 pontos) Prove ou refute: para todo grafo não-direcionado e conexo $G = (V, E)$ com arestas ponderadas por uma função $w : E \to \mathbb{R}$, se uma certa aresta $\{u, v\} \in E$ possui o menor peso dentre todas as arestas de G que incidem em u, então existe uma árvore geradora mínima de G que possui a aresta $\{u, v\}$. Observação: consulte o professor sobre o uso de teoremas anteriores. Dica: é possível argumentar de forma parecida à que fizemos para o algoritmo de Kruskal.

Demonstração. Seja $T = (V, F)$ uma AGM de G. Se $\{u, v\} \in F$, então T é uma testemunha do resultado, CQD. Suponhamos agora que $\{u, v\} \notin F$. Nesse caso, como u e v são vértices de T e T é um grafo conexo, existe um caminho p de u a v em T. Seja então $\{u, x\}$ a primeira aresta de p; obviamente, $x \neq v$. Observe que, pela hipótese sobre a aresta $\{u, v\}$, temos $w(u, v) \leq w(u, x)$. Logo, sendo $F' = (F \setminus \{u, x\}) \cup \{u, v\}$ e $T' = (V, F')$, temos $w(T') \leq w(T)$. Assim, para mostrarmos que T' é uma AGM de G, resta apenas mostrar que T' é uma árvore (já que T' claramente é subgrafo gerador de G). De fato, observe que o grafo $T'' = (V, F \setminus \{u, x\})$ possui exatamente duas componentes conexas, pois T'' é igual à arvore T com uma aresta a menos. Além disso, u e v estão em componentes conexas distintas em T'', pois p é o único caminho entre u e v em T, e, como $\{u, x\}$ – que é uma aresta de p – não está em T'', então não existe caminho entre u e v em T''. Logo, como T' consiste em T'' – que é acíclico – adicionado de uma aresta que liga vértices das diferentes componentes de T'', então T' é conexo e acíclico, ou seja, uma árvore. Logo, T' é uma testemunha do enunciado, CQD. □