MC448: Projeto e Análise de Algoritmos I

Profs. Cid C. de Souza e Ricardo Dahab – 2ª Prova – (27/05/2008)

Nome:	
RA:	**Turma:**

Observação: o peso das questões será decidido pelo docente da seguinte forma: a questão que você responder melhor terá peso 4 e as duas outras terão peso 3. Portanto, uma mesma questão pode ter peso 3 para um aluno e peso 4 para um outro aluno.

Questão	frac	Peso	Nota
1			
2			
3			
Total		10,0	

Instruções: A duração da prova é de 105 minutos. Não é permitido usar qualquer material de consulta durante a prova. **Questões mal justificadas serão consideradas erradas !** O termo **ordenado** é usado aqui para denotar vetores ou seqüências em **ordenação não decrescente**.

1. Seja S um conjunto de n elementos e X o vetor **ordenado** contendo os elementos de S. Os elementos de um conjunto de $k-1$ estatísticas de ordem que dividem X em em k subvetores de mesmo tamanho **a menos de uma unidade** (ou seja, com tamanhos diferindo por no máximo um) são chamados de k-ésimos *quantis* de S .
Abaixo são dados alguns exemplos para diferentes valores de n e k. Os k-ésimos quantis estão indicados em negrito e sublinhados. Note pelos exemplos que um mesmo vetor pode apresentar diferentes conjuntos de quantis

$n=10$	1	2	3	4	5	6	7	8	9	10
$k=3$	10	12	17	**22**	23	29	**33**	42	57	61

$n=14$	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$k=4$	10	12	**17**	22	23	**29**	33	42	57	**61**	70	72	74	81

$n=14$	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$k=4$	10	12	17	**22**	23	29	**33**	42	57	61	**70**	72	74	81

Seja q o vetor ordenado de tamanho $k-1$ que armazena as posições de um conjunto de k-ésimos quantis de S (p.ex., para o primeiro caso acima, teríamos $q=[4,7]$). Responda os itens abaixo:

(a) mostre que os subvetores associados aos k-ésimos quantis não podem ter tamanhos menores que $\lfloor \frac{n}{k} \rfloor$ e nem maiores que $\lceil \frac{n}{k} \rceil$.

(b) tendo em vista o resultado do item anterior, mostre como computar em $O(1)$ o número de subvetores de tamanho $\lfloor \frac{n}{k} \rfloor$ e $\lceil \frac{n}{k} \rceil$ em que os k-ésimos quantis dividem o vetor ordenado.

(c) usando os resultados do item anterior, mostre em pseudo-código como você calcularia os valores do vetor q definido acima.

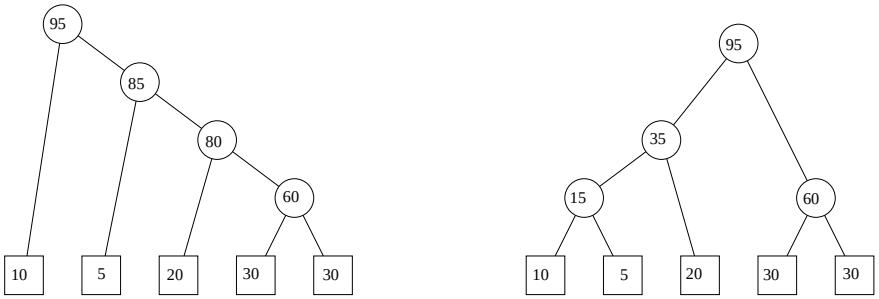
(d) Suponha que sejam conhecidas as **posições** das $k-1$ estatísticas que devam ser calculadas (isso foi feito nos itens anteriores). Descreva um algoritmo de complexidade $O(n \log k)$ que lista os valores dos k-ésimos quantis de S. Você pode usar como subrotinas os procedimentos de cálculos de estatísticas de ordem vistos em sala de aula.

Não deixe de justificar porquê a complexidade é aquela solicitada.

2. Suponha que você dispõe de um procedimento INTERCALA2 que recebe como entrada dois vetores **ordenados** X e Y de números inteiros de tamanhos x e y, respectivamente, retornando na saída um vetor Z **ordenado** de tamanho $x+y$. Você sabe que este algoritmo executará $\Theta(x+y)$operações no pior caso.

Suponha agora que você receba os n vetores **ordenados** de números inteiros $A_1, \ldots, A_n$ de tamanhos $t_1, \ldots, t_n$, respectivamente. Você deseja criar o vetor **ordenado** B contendo todos os elementos dos n vetores passados na entrada. Para isso, só pode fazer intercalação (*merge*) de vetores **aos pares** usando o procedimento INTERCALA2.

Para uma dada entrada, existem várias formas de você fazer as intercalações. Como elas são feitas aos pares, podemos representar o processo usando **árvores binárias**. Por exemplo: considere a entrada formada por 5 vetores de tamanhos $10, 5, 20, 30$ e 30. Abaixo são mostrados duas formas distintas de efetuar as intercalações. Note que no interior de cada nó interno, encontra-se a ordem do número total de operações realizadas no pior caso na intercalação (execução de INTERCALA2) representada pelo nó. Os nós folhas representam os vetores originais e os números dentro de cada um deles representam o tamanho do vetor correspondente. Em cada caso, o número **total** de intercalações no pior caso corresponde à soma dos valores contidos nos nós internos, ou seja, 320 e 205 para as árvore esquerda e direita, respectivamente.



Suponha que você deseja minimizar o número **total** de intercalações no pior caso para construir o vetor B. No caso do exemplo acima, as intercalações representadas pela árvore da direita levam a este mínimo.

Projete um **algoritmo guloso** de complexidade $O(n \log n)$ que encontre este valor mínimo, respondendo aos seguintes itens:

(a) enuncie o critério que corresponde à escolha gulosa.

(b) escreva um pseudo-código em alto-nível do seu algoritmo, explicando as eventuais estrutruras de dados que você usou.

(c) mostre que a complexidade é de fato $O(n \log n)$.

(d) dê os argumentos que provam a corretude do seu algoritmo.

3. Você recebeu um tabuleiro de xadrez de dimensão $n \times n$ e um peão. Considere que as linhas (colunas) do tabuleiro estão numeradas de 1 a n começando de baixo (da esquerda). Você deve mover o peão de linha 1 (mais baixa) para a linha n (mais alta) do tabuleiro de acordo com as seguintes regras. A cada passo você pode mover o peão para uma das seguintes casas (em relação à casa onde ele está atualmente):

1. A casa imediatamente acima.
2. A casa uma linha acima e uma coluna à esquerda, desde que o peão não se encontre atualmente na coluna mais à esquerda (número 1).
3. A casa uma linha acima e uma coluna à direita, desde que o peão não se encontre atualmente na coluna mais à direita (número n).

Se você mover o peão de uma casa x para uma casa y, você recebe um prêmio $p(x, y)$, não necessariamente positivo.
O problema que se deseja resolver é encontrar uma seqüência de movimentos válidos no tabuleiro que move o peão de *alguma* casa da linha 1 para *alguma* casa na linha n e que **maximize** o total de prêmios recebidos. Note que você é livre para decidir em que casa o peão sai da linha de baixo (1) e chega na linha de cima (n).
Você deve projetar um algoritmo de **programação dinâmica** para este problema. Para isso, defina a matrix quadrada g de dimensão n onde $g[i, j]$ representa o valor da maior premiação que pode ser recebida saindo-se de alguma casa da linha 1 (mais baixa) e chegando-se na casa da linha i e coluna j. Responda então aos seguintes itens:

(a) mostre que este problema tem subestrutura ótima.

(b) usando a definição de g, dê a *fórmula de recorrência* que reflete a existência da subestrutura ótima.

(c) quais células de g você pode inicializar com valores triviais e quais são estes valores.

(d) mostre por um diagrama qual seria uma possível ordem de preenchimento de g para que a fórmula de recorrência do item *(b)* retorne valores corretos.

(e) escreva o pseudo-código do algoritmo de programação dinâmica que resolve o problema. **Nota:** você pode assumir que o algoritmo retornará apenas valor da maior premiação.

(f) calcule a complexidade do seu algoritmo.