

Terceiro teste de Algoritmos e Estruturas de Dados

9 de Dezembro de 2019

14h10m – 15h00m

Responda a todas as perguntas no enunciado do teste. Justifique todas as suas respostas.
O teste é composto por 5 grupos de perguntas.

Nome: _____

N. Mec.: _____

4.0 **1:** O algoritmo *merge sort* divide o *array* a ser ordenado ao meio, ordena (recursivamente) cada uma das duas partes, e depois junta-as. A sua complexidade computacional é $\Theta(n \log n)$. Um aluno está convencido que se em vez de se dividir o array em duas partes se se dividir em três partes (todas mais ou menos do mesmo tamanho), então a complexidade computacional desta variante do *merge sort* será ainda mais baixa. Responda às seguintes perguntas:

- 1.0 a) Que estratégia algorítmica usa o *merge sort*?
- 2.0 b) O aluno tem razão? Justifique.
- 1.0 c) Indique uma desvantagem do *merge sort*, quando comparado com o *quicksort*.

Respostas:

O *master theorem* afirma que se $T(n) = aT(n/b) + f(n)$ então

- se $f(n) = O(n^{\log_b a - \epsilon})$ para um $\epsilon > 0$, então $T(n) = \Theta(n^{\log_b a})$,
- se $f(n) = \Theta(n^{\log_b a})$, então $T(n) = O(n^{\log_b a} \log n)$,
- se $f(n) = \Omega(n^{\log_b a + \epsilon})$ para um $\epsilon > 0$ e se $af(\frac{n}{b}) \leq cf(n)$ para $c < 1$ e n suficientemente grande, então $T(n) = \Theta(f(n))$.

- 6.0 **2:** Num tabuleiro de xadrez, pretende-se ir do canto inferior esquerdo $(0, 0)$ para o canto superior direito $(7, 7)$ fazendo movimentos apenas para a direita e para cima. Quando se está em (x, y) , o custo de ir para a direita é dado por R_{xy} e o custo de ir para cima é dado por U_{xy} . O custo total é a soma dos custos dos 14 movimentos efetuados (7 para a direita e 7 para cima). Sabe-se que $4 \leq R_{xy} \leq 10$ e que $6 \leq U_{xy} \leq 20$. No programa que foi usado para calcular o **custo mínimo** para ir de $(0, 0)$ até $(7, 7)$, os valores de R_{xy} e U_{xy} estão guardados nas matrizes

```
int R[7][8], U[8][7]; // initialized elsewhere
```

Responda às seguintes perguntas:

- 3.0 a) Complete o seguinte código, que resolve o problema usando a técnica *branch-and-bound*.

```
void go_to(int x,int y,int partial_cost,int *min_cost) {  
    if(partial_cost +  * (7 - x) + 6 * ()  *min_cost) return;  
    if(x < 7) go_to(x + 1,y,partial_cost + ,min_cost);  
    if(y < 7) go_to(x,y + 1,partial_cost + ,min_cost);  
    if(x == 7 && y == 7) *min_cost = partial_cost;  
}  
  
int compute_min_cost_bb(void) {  
    int min_cost = ; go_to(0,0,,&min_cost); return min_cost; }
```

- 3.0 b) Pretende-se também resolver este problema usando programação dinâmica. Para isso, o custo mínimo para ir de $(0, 0)$ até (x, y) é guardado na matriz

```
int C[8][8];
```

Complete o seguinte código. (No fim, estamos interessados no valor de $C[7][7]$, mas, para o calcular, dá jeito conhecer os outros valores.)

```
int update_C(int x,int y,int C[8][8]) {  
    if(x  0 || y < 0) return 1000000000;  
    if(C[x][y]  0)  
    {  
        int Cx = update_C(x - 1,y,C) + ;  
        int Cy = update_C(x,y - 1,C) + ;  
        C[x][y] = (Cx < Cy) ? Cx : Cy;  
    }  
    return C[x][y]; }  
  
int compute_min_cost_dp(void) {  
    int C[8][8]; // -1 means not yet known  
    for(int x = 0;x < 8;x++)  
        for(int y = 0;y < ;y++)  
            C[x][y] = (x == 0 && y == 0) ? 0 : ;  
    update_C(,,C);  
    return C[7][7]; }
```

3.0 **3:** Os 5 vértices, numerados de 0 a 4, de um grafo têm as seguintes listas de adjacência:

0	→ 1 → 3 → 4 → NULL
1	→ 4 → 2 → NULL
2	→ NULL
3	→ 2 → NULL
4	→ 3 → 1 → NULL

Responda às seguintes perguntas:

- 1.0 **a)** Desenhe o grafo.
- 1.0 **b)** Represente o grafo usando uma matriz de adjacência.
- 1.0 **c)** É possível representar este grafo usando um único inteiro de **32 bits**? Se sim, como? Se não, por que não?

Respostas:

3.0 **4:** Explique para que serve e como funciona o algoritmo *union find*.

- 4.0 **5:** Considere um labirinto desenhado na superfície de uma esfera. Pretende-se ir, a andar, do pólo norte até ao pólo sul. (Considere que a esfera tem um raio relativamente pequeno, pelo que ir a pé não demora meses, mas pode demorar dias.) Responda às seguintes perguntas:
- 1.0 a) Que algoritmo usaria para encontrar uma solução?
 - 1.5 b) Que material levaria consigo para o ajudar?
 - 1.5 c) Acha que é possível encontrar a solução mais curta de uma forma eficiente? (Não se esqueça que tem de fazer todo o caminho a pé.) Porquê?

Answers: