ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

Final Exam 4 July 2006

Algorithmics

4 July 2006

- The only document allowed is a 2-sided $A4$ sheet of notes.
- No calculators, cell phones, laptops, etc...
- Write your answers directly on this document. If you need extra sheets, they must all contain your name and the problem number on the top of the page.
- Do not answer two questions on the same page.

Last name :

First name :

Section :

Exercise 1 /15 points	Exercise 2 /15 pts	Exercise 3 /15 pts	Exercise 4 /15 pts	Exercise 5 /15 pts	Exercise 6 /18 pts	Exercise 7 /15 pts	Exercise 8 /18 pts

Total

Problem 1 [15 points].

Consider the following problem: Given a (not necessarily sorted) sequence of n integers $a[0], \ldots, a[n-1]$, determine whether there are two indices i and j with

$$a[i] = 2 \cdot a[j].$$

a) Give a formal specification of this problem.

b) Describe an algorithm that solves this problem using $O(n \cdot \log n)$ operations.
(*Hint:* Use algorithms we have seen in the lectures).

Problem 2 [15 points].

Suppose we have the following instance of the 0/1-Knapsack problem: We want to maximize the total value by choosing a subset of the objects $\{A_1, \ldots, A_7\}$, with a maximum weight of $W = 20$.

Object	A_1	A_2	A_3	A_4	A_5	A_6	A_7
Value	3	7	12	11	14	6	x
Weight	2	4	6	8	11	4	y

Recall that the algorithm seen in lectures computes a matrix C with dimensions $(n+1) \times (W+1)$ (so 8×21 in this case).

The 6 first lines of C are given below:

$$\begin{pmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 \\
0 & 0 & 3 & 3 & 7 & 7 & 10 & 10 & 10 & 10 & 10 & 10 & 10 & 10 & 10 & 10 & 10 & 10 & 10 & 10 & 10 \\
0 & 0 & 3 & 3 & 7 & 7 & 12 & 12 & 15 & 15 & 19 & 19 & 22 & 22 & 22 & 22 & 22 & 22 & 22 & 22 & 22 \\
0 & 0 & 3 & 3 & 7 & 7 & 12 & 12 & 15 & 15 & 19 & 19 & 22 & 22 & 23 & 23 & 26 & 26 & 30 & 30 & 33 \\
0 & 0 & 3 & 3 & 7 & 7 & 12 & 12 & 15 & 15 & 19 & 19 & 22 & 22 & 23 & 23 & 26 & 26 & 30 & 30 & 33 \\
? & ? \\
? & ?
\end{pmatrix}.$$

So the lines corresponding to A_6 and A_7 have not yet been computed.

a) If $x = 6$ and $y = 3$, what is the solution to this problem? Give the biggest value achievable, and a subset of the objects that achieves this value (without going over the maximum weight).

b) If $y = 7$, express the optimal value as a function of x.

c) If $x = 11$, give a value of y for which there are two optimal solutions (two distinct subsets that both give the greatest total value).

We did not cover knapsack in this year's course. But the recursion that defines the DP is

$$K[i,W] = \max \left\{ K[i-1,W],\ K[i-1, W-\text{weight}(A_i)] + \text{value}(A_i) \right\}$$

$$K[0,W] = K[i,0] = 0$$

$K[i, w]$ stands for

"profit we can obtain by only packing items among the i first in a knapsack of capacity w"

Therefore the recursion

$$K[i,w] = \max \left\{ \underbrace{K[i-1, w]}_{\text{item } i \text{ was not picked in solution}} ,\ \underbrace{K[i-1, w-\text{weight}(A_i)] + \text{value}(A_i)}_{\text{item } i \text{ was picked in solution}} \right\}$$

Problem 3 [15 points].

Consider the probability distribution $\{\frac{1}{34}, \frac{1}{34}, \frac{1}{34}, \frac{2}{34}, \frac{3}{34}, \frac{5}{34}, \frac{8}{34}, \frac{13}{34}\}$.

a) Construct a Huffman code for this distribution.

b) Calculate the average codeword length.

c) Give codewords of an optimal code with shortest codeword of maximum length.

d) Give codewords of an optimal code with longest codeword of maximum length.

Huffman Codes were not covered

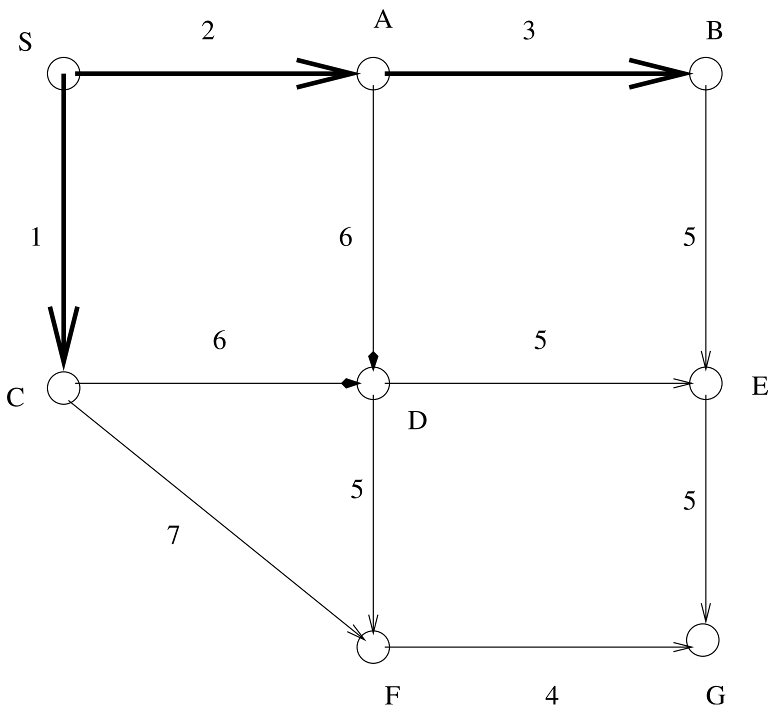
Problem 4 [15 points]. We are running each one of the following algorithms on the graph below, where the algorithm has already processed (ignore the directions on the edges for Prim's and Kruskal's) the bold-faced edges : Prim, kruskal and Dikstra. ← not corrected

1. Which edge would be added next in Prim's algorithm. Justify your answer.

2. Which edge would be added next in Kruskal's algorithm ? Justify your answer.

3. Which vertex would be added in T at the next iteration in Dijkstra's algorithm, if $T = \{S, A, B, C\}$? Justify your answer.

4. Which vertex would be added in T at the final iteration in Dijkstra's algorithm ? Justify your answer.



Problem 5 [15 points]. A computer network is modeled as a directed graph $G = (V, E)$ whose nodes correspond to routers and edges represent the links between them. So a data packet can travel from a router to one of its (directed) neighbors in one step. Assume that each network link e_i is alive with some positive probability p_i and down with probability $1 - p_i$, independently of the other links. We want to send a packet through a single path from node s to node t. We know that there is at least one directed path from s to t.

a) Suppose that there is a path of length ℓ from s to t along the edges $e_1, \ldots, e_\ell$. What is the probability of this path being alive? ("being alive" obviously means that all the links along the path are working.)

b) Now among all possible paths we want to choose one that has maximum probability of being alive. Design an efficient algorithm (with a running time of at most $O(|V|^3)$) for finding such a path. (*Hint:* How does the probability in (a) evolve with the length of the path?)

Problem 6 [18 points]. Recall the procedure that implements quicksort algorithm:

```
1: QuickSort($A, p, r$)
2: if $p < r$ then
3:    $q \leftarrow$ Partition($A, p, r$)
4:    QuickSort($A, p, q - 1$)
5:    QuickSort($A, q + 1, r$)
6: end if
```

The procedure Partition(A, ℓ, r) takes the pivot piv $= a[r]$, permutes the elements of the list and returns an index q such that

$$\begin{aligned} a[q] &= \text{piv} \\ q < i \leq r \Longrightarrow a[i] &> \text{piv} \\ \ell \leq i < q \Longrightarrow a[i] &\leq \text{piv}. \end{aligned}$$

Its running time is $O(n)$, where n is the size of the list, i.e., $n = r - \ell + 1$.

Now consider the following slightly different procedure (i is a positive integer between 1 and n):

```
1: Modified($A, p, r, i$)
2: if $p = r$ then
3:    return $A[p]$
4: end if
5: $q \leftarrow$ Partition($A, p, r$)
6: $k \leftarrow q - p + 1$
7: if $i = k$ then
8:    return $A[q]$
9: else if $i < k$ then
10:    return Modified($A, p, q - 1, i$)
11: else
12:    return Modified($A, q + 1, r, i - k$)
13: end if
```

1. What does Modified do? Justify.
2. Analyze the asymptotic running time of Modified in the best and worst cases (assuming that Partition always takes the last given element as the pivot).

Problem 7 [15 points]. Show that the following decision problem is NP-complete: given a graph $G = (V, E)$ and an integer ℓ, decide if there exists a path of length ℓ in G which doesn't visit any vertex twice.

Hint: Recall that the problem to decide if a graph contains a Hamiltonian path (i.e., a path visiting each vertex of the graph exactly once) is NP-complete. Use this result.

Not covered in this year's course

Problem 8 [18 points]. Recall that in a binary tree T, the *balance factor* of a node is defined as

$$\mathrm{Bal}(x) = h(T_1) - h(T_2),$$

where T_1 and T_2 are respectively the left and right subtrees of x. A tree is said to be *balanced* if

$$\forall x \in V(T): \quad \mathrm{Bal}(x) \in \{-1, 0, 1\}.$$

A tree is said to be *weekly balanced* if

$$\forall x \in V(T): \quad \mathrm{Bal}(x) \in \{-2, -1, 0, 1, 2\}.$$

Let X_h be the minimal number of nodes in a weekly balanced tree of height h. So we have for example $X_0 = 1$, $X_1 = 2$ et $X_2 = 3$.

a) Find values a, b, c and d for which for all $h \geq 2$:

$$X_{h+1} = a \cdot X_h + b \cdot X_{h-1} + c \cdot X_{h-2} + d.$$

b) Show by induction that for all $h \geq 0$ we have

$$X_h \leq 8 \cdot \left(\frac{3}{2}\right)^h.$$

We didn't talk about balanced trees but this exercise is doable nevertheless.