Faculty of Science
Final Examination

Computer Science COMP-251B
Data Structures and Algorithms

Examiner: Prof. Claude Crépeau **Date:** April 18, 2011
Associate Examiner: Prof. Clark Verbrugge **Time:** 14:00 – 17:00

INSTRUCTIONS:

This examination is worth 50% of your final grade.
The total of all questions is 100 points.
Each question is assigned a value found in brackets next to it.

O P E N • B O O K S •/• O P E N • N O T E S

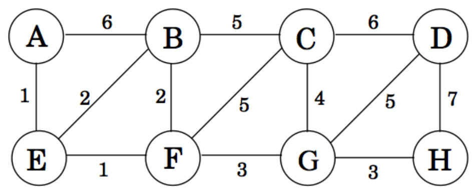
Faculty standard calculator permitted only.
This examination consists of 4 pages including title page.
This examination consists of 6 questions.

SUGGESTION : read all the questions and their values *before* you start.

1)

[15%]

Consider the following graph.



(a) What is the cost of its minimum spanning tree?

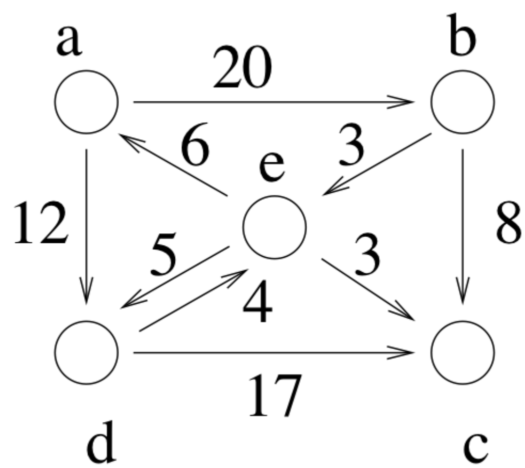
(b) How many minimum spanning trees does it have?

(c) Suppose Kruskal's algorithm is run on this graph. In what order are the edges added to the MST? For each edge in this sequence, give a cut that justifies its addition.

2)

[15%]

Consider the following directed graph:



Using the Ford-Fulkerson algorithm, find the maximum flow from node **(a)** to node **(c)** in this graph.

A. For each loop of the algorithm, provide the flow constructed so far as well as the residual graph.

B. Provide explicitly a criterion that you used to decide which of several paths will be considered first. This is up to you, many answers may be valid here.

C. Upon termination, provide a minimum cut of the vertices. How did you find it ?

[10%]

3) TRUE and FALSE…

For each statement, say if it is *true* or *false*.

Correct = +1 pt, Incorrect = -0.5 pt, No answer = 0 pt, Minimum Total= 0 pt.

(a) Multiplying two **n**-bit integers, requires to run at least $\mathbf{\Omega(n)}$ time in the worst case.
(b) Finding the two closest points (out of **n**) in a plane, may be done in time $\mathbf{O(n)}$.
(c) The hash function below is universal.

```
int h(String s, int n) {

   int hash = 0;

   for (int i = 0; i < s.length();
i++)

      hash = (31 * hash) + s[i];

   return hash % n;

}
```

(d) Deterministic algorithms are always as efficient as probabilistic algorithms.
(e) The problem of interval partitioning is solved efficiently using divide & conquer.
(f) The Bellman-Fulkerson algorithm can deal with negative edges in shortest paths.
(g) The number **N!** is a $\mathbf{\Theta(N \log N)}$-bit long integer.
(h) The min cut in a directed graph is unique.
(i) Dijkstra's algorithm is implemented using the Disjoint-set Data structure.
(j) The expected number of comparisons in randomized quicksort is less than **n ln n**.

[25%]

4) SHORT and SWEETS…

NHL Eastern Conference

As of April 3, 2011, at 09:13 AM ET

GP = Games Played
W = Games Won (worth 2 points)
L = Games Lost (regular) (worth 0 points)
OT = Games lost in Over Time (worth 1 point)
GL = Games Left to play
(TEAM name if in list, or "$" if unlisted team)

PTS = 2xW + OT

RANK	TEAM	GP	W	L	OT	GL	PTS
1	Washington	79	46	22	11	TOR, $, $	103
2	Philadelphia	78	46	22	10	NYR, $,BUF, $	102
3	Boston	78	44	23	11	NYR, $, $, $	99
4	Pittsburgh	79	46	25	8	$, $, $	100
5	Tampa Bay	78	43	24	11	$,BUF, $,CAR	97
6	Montreal	79	42	30	7	$, $,TOR	91
7	Buffalo	78	39	29	10	CAR,TB,PHI, $	88
8	NY Rangers	78	41	32	5	PHI,BOS, $, $	87
9	Carolina	78	38	30	10	BUF, $,TB	86
10	Toronto	79	37	32	10	WAS, $,MTL	84

(a) For each of the 10 first teams in the Eastern Conference of the **National Hockey League**, give a proof as whether the team may or not still finish first in its conference.

(b) Why not use memoization in all recursive algorithms ?

(c) Consider a general algorithm to find shortest paths in directed graphs that does not deal with negative edges. If you have a graph with negative edges, why not add a large positive constant to every edge and then solve the resulting graph ?

(d) Is it possible to design an algorithm that is greedy, divide-and-conquer and dynamic programming all at the same time? Why ??

5)

[15%]

Minimizing Lateness: Greedy Algorithms

Greedy template. Consider jobs in some order.

- [Shortest processing time first] Consider jobs in ascending order of processing time t_j.

	1	2
t_j	1	10
d_j	100	10

counterexample

- [Smallest slack] Consider jobs in ascending order of slack $d_j - t_j$.

	1	2
t_j	10	1
d_j	10	2

counterexample

Consider this page from the course slides. Explain these two counter-examples: These are counter-examples to what ? and why are they counter-examples…?

6)

[20%]

Given two strings $x = x_1x_2\cdots x_n$ and $y = y_1y_2\cdots y_m$, we wish to find the length of their *longest common substring*, that is, the largest k for which there are indices i and j with $x_ix_{i+1}\cdots x_{i+k-1} = y_jy_{j+1}\cdots y_{j+k-1}$. Show how to do this in time $O(mn)$.

a) Show how to find the length of the longest common suffix **LCSuf[p,q]** (**p ≤ n** and **q ≤ m**) for all pairs of prefixes $x_{1..p}$, $\mathbf{y}_{1..q}$ using a simple recursion between **LCSuf[p,q]** and **LCSuf[p-1,q-1]**.

b) Given all the longest common-suffixes for all pairs of prefixes of the strings, how do we find the longest common-substring **LCSub[n,m]** ?

c) Write both an iterative and a recursive (with memoization) algorithm to compute the longest common substring of **x** and **y** using the results of **a)** and **b)**.

d) Show that the running time of one of your algorithms from **c)** is **O(mn)**.