Faculty of Science
Final Examination

Computer Science COMP-251A
Data Structures and Algorithms

Examiner:	Prof. Claude Crépeau	**Date:**	Dec. 15, 2014
Associate Examiner:	Prof. Prakash Panangaden	**Time:**	9:00 – 12:00

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

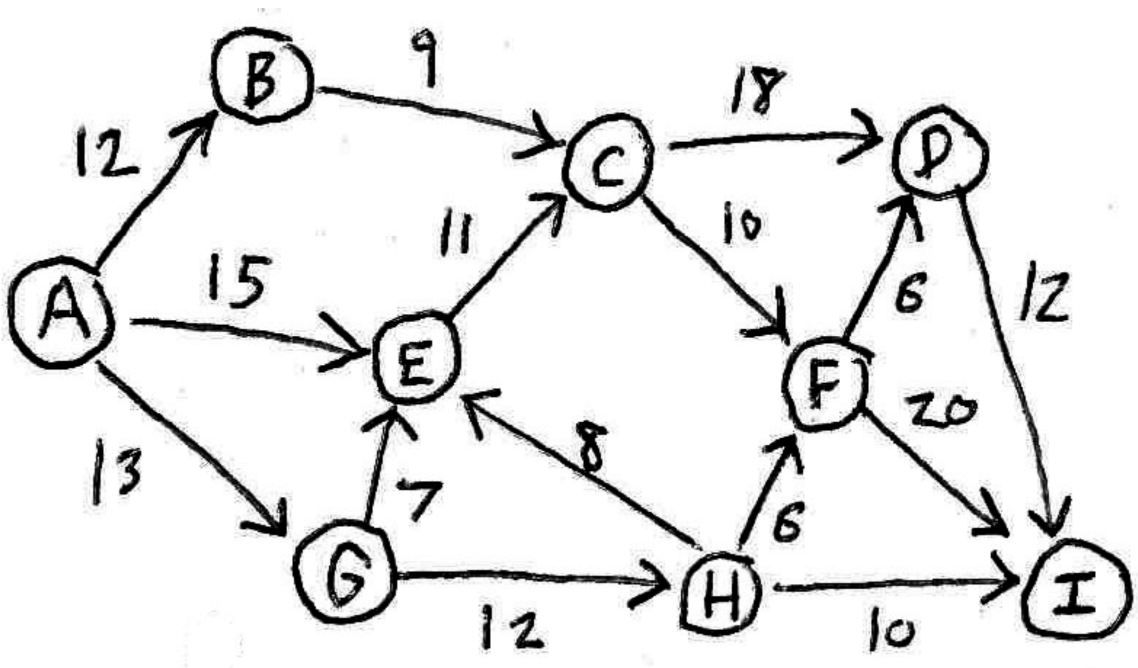
- Faculty standard calculator permitted only.
- This examination consists of 5 pages including title page.
- This examination consists of 5 questions.

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

[15%]

1) Ford-Fulkerson

Consider the following directed graph



Using the Ford-Fulkerson algorithm, find the maximum flow from **(A)** to **(I)** 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 you 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 ?

2) The coin game.

[25%]

Consider a row of n coins of values **v(1) ... v(n)**, where **n** is even. We play a game against an opponent by alternating turns. In each turn, a player selects either the first or last coin from the remaining row, removes it from the row permanently, and adds the value of the coin to his previous wins. Determine the maximum possible amount of money we can definitely win if we move first.

a) Give a greedy solution to this problem and show that it is not optimal.

Define **V(i,j)** as the maximum we can definitely win (even if the opponent plays as well as possible) when it is our turn to play and only coins **v(i),…,v(j)** remain (**j–i** is odd).

b) For each index **i,** what is **V(i,i+1)** ?

c) Find a recursive formula for **V(i,j)** from (four) other values of **V(i',j')** (**j'–i'** is odd).
Hint: When **v(i),…,v(j)** remain, only **v(i)** or **v(j)** can be selected. Notice also that we are trying to maximize our win whereas the opponent is trying to minimize our win.

d) Write both an iterative and a recursive (with memoization) algorithm to compute the value **V(1,n)** using the results in **b)** and **c)**.

e) Show that the running time of **one** of your algorithms in **d)** is **O(n^2)**.

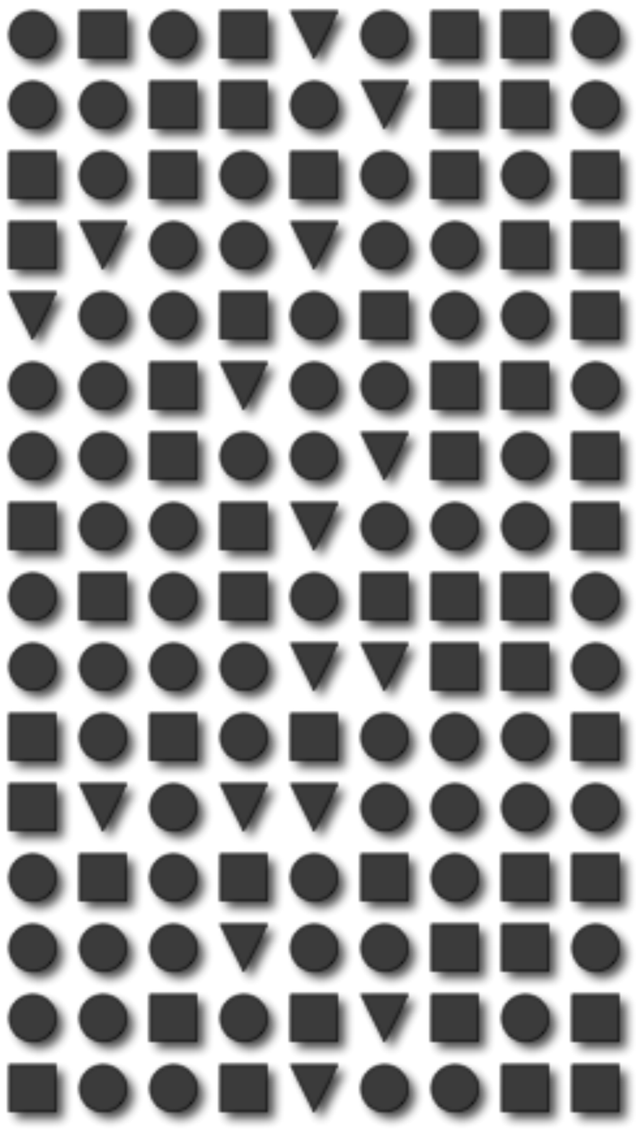
[25%]

3) Majority.

You are given an array **A[1..n]** of integers. You would like to determine whether **A** contains a ***majority element***, that is, an element that appears more than **n/2** times in **A**.

a) Use sorting to provide an **O(n log n)** time algorithm to solve this problem.

Let's now consider a similar situation but with data *that cannot be sorted* ! You are given an array **A[1..n]** of general objects. The only comparison that we have on these objects is equality between two elements, no ``**<**´´ or ``**>**´´ used for sorting… The picture to the right is an example of objects that might not have ``**<**´´ or ``**>**´´ in their toolbox…



b) Show that if **x** is a candidate object from array **A[1..n]** you can decide in **O(n)** time whether **x** is (or not) a majority element of **A** using only equality tests.

c) Now notice that if you have a majority element in **A[1..n]** then it must be the majority element in either **A[1..$\lfloor$n/2$\rfloor$]** or **A[$\lfloor$n/2$\rfloor$+1..n]** as well. Use this fact and the algorithm in **b)** to design a divide-and-conquer algorithm for solving this problem.

d) Establish a recurrence relation for the running time of the algorithm in **c)** and determine its solution using the master method.

e) Finally, consider the following short algorithm:

```
count:=0;
FOR i:=1 TO n DO
    IF count=0 THEN candidate:=A[i]; count:=1
                ELSE IF candidate=A[i] THEN count:=count+1
                                       ELSE count:=count –1
```

This algorithm has the wonderful property that

``If **A** contains a majority element then the final value of *candidate* is this element´´

(I am not asking you to prove this). Combine with a previous part to get an **O(n)** time algorithm for the problem of finding a majority element using only equality tests…

[10%]

4) True or False.

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

Correct = +1 pt, Incorrect = -1/2 pt, No answer = 0 pt, Minimum Total= 0 pt.

(a) Multiplying two **nxn** binary matrices, requires at least $\Omega(n^2)$ time in worst case.

(b) Any comparison-based sorting cannot possibly be correct on all inputs of size **n** if the time it takes **is not** $\Omega(n \log n)$.

(c) The **Ford** of Bellman-**Ford** and **Ford**-Fulkerson is the same and unique person.

(d) In Radix sort it is mandatory that the underlying sorting algorithm be *stable*.

(e) A tree is a forest and a forest is a tree.

(f) In a **red-black** tree, at least one child of a **red** node must be **red**.

(g) It is always possible to sort **n** elements in the best-case time O(n).

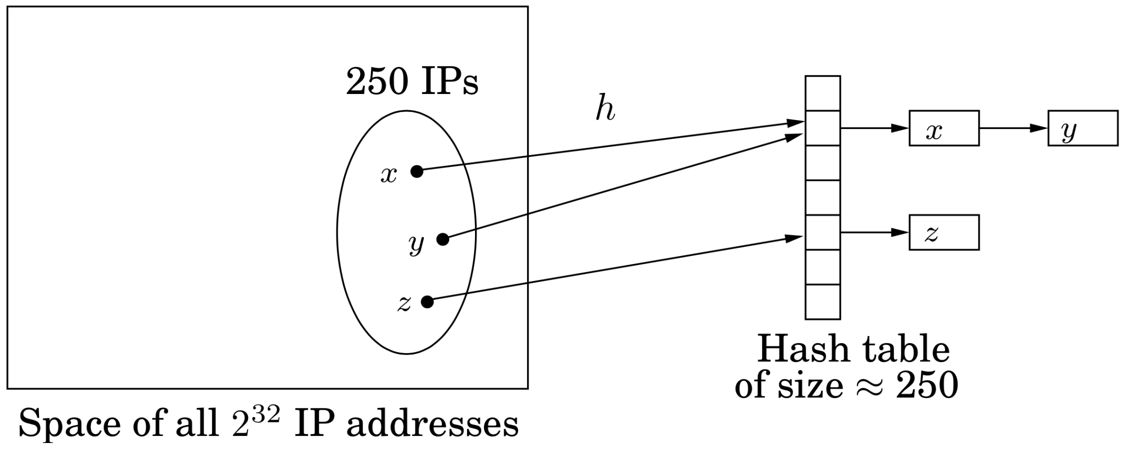
(h) When a graph contains negative edges, the length of the shortest path is undefined.

(i) Probabilistic algorithms are always more efficient than deterministic algorithms.

(j) The number **N!** is a $\Theta(N \log N)$-bit long integer.

[25%] **5) Hash Functions.**

An IP address is a 4-tuple of numbers $\mathbf{x=x_1.x_2.x_3.x_4}$ with $\mathbf{0 \le x_i \le 255}$. For example, **crypto.cs.mcgill.ca** (the server of my research group where the course web page is located) has IP address **132.206.3.14**. On every computer connected to the internet, address **127.0.0.1** refers to the computer itself. There are $\mathbf{2^{32}}$ possible IP addresses. In this problem we want to store roughly **250** IP addresses from our costumers in a hash table.



Let $\mathbf{H_{256}} = \{ \mathbf{h_a} : \mathbf{a} \in \{\text{all } \mathbf{4}\text{-digit, base-}\mathbf{256} \text{ integers}\} \}$.

a) If we randomly select a hash function from $\mathbf{H_{256}}$, show that there exist IP addresses **x** and **y** such that $\mathbf{Pr_a[h_a(x) = h_a(y)] > {}^1/_{256}}$.
Hint: 2k mod 256 is never odd.

Let $\mathbf{H_{257}} = \{ \mathbf{h_a} : \mathbf{a} \in \{\text{all } \mathbf{4}\text{-digit, base-}\mathbf{257} \text{ integers}\} \}$ (**257** is a prime).

b) If $\mathbf{h_a}$ is randomly selected from $\mathbf{H_{257}}$, what is the expected number of collisions in the **257**-entry table for the **250** values we have to store ?

c) If we select $\mathbf{h_a}$, where **a=(2,47,199,256)**, from $\mathbf{H_{257}}$ what is the value of $\mathbf{h_a}$**(127.0.0.1)** ?

d) Find a collision for the value **127.0.0.1** under $\mathbf{h_a}$, **a=(2,47,199,256)**.

e) The hash function $\mathbf{h_a}$, where **a=(0,0,0,0)**, is rather peculiar because it maps every IP address to **0** despite the fact that **257** is prime. Wouldn't we be better off selecting from $\mathbf{H'_{257}} = \{ \mathbf{h_a} : \mathbf{a} \in \{\text{all } \mathbf{4}\text{-digit, base-}\mathbf{257} \text{ integers, except } \mathbf{(0,0,0,0)}\} \}$ instead of $\mathbf{H_{257}}$? Explain.