NATIONAL UNIVERSITY OF SINGAPORE

CS4234 - Optimization Algorithms

(Semester 1 AY2015/16)

Time Allowed: 2 Hour

INSTRUCTIONS TO STUDENTS

a. Write your Student Number below, and on every page. Do not write your name.

b. The assessment contains **SIX** multi-part problems (and one just for fun). You have **120** minutes to earn **100 points**.

c. The assessment contains 18 pages, including the cover page and 4 pages of scratch paper.

d. The assessment is closed book. You may bring two double-sided sheet of A4 paper to the assessment. You may not use a calculator, your mobile phone, or any other electronic device.

e. Write your solutions in the space provided. If you need more space, please use the scratch paper at the end of the assessment. Do not put part of the answer to one problem on a page for another problem.

f. Show your work. Partial credit will be given. Please be neat.

g. You may use (unchanged) any algorithm or data structure that we have studied in class, without restating it. If you want to modify the algorithm, you must explain exactly how your version works.

h. Draw pictures frequently to illustrate your ideas.

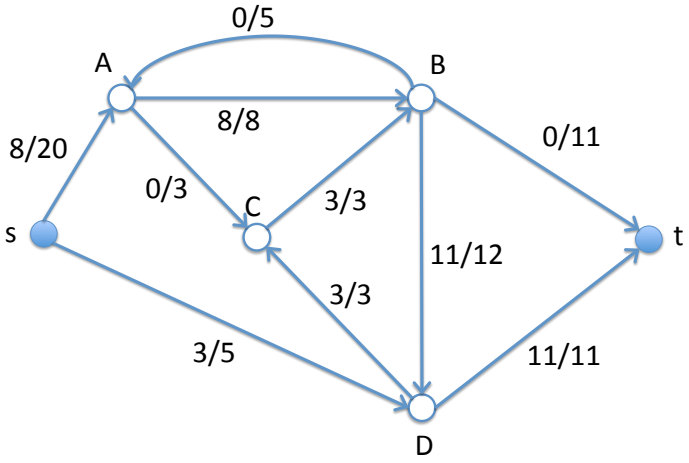
i. Good luck!

Student Number: _____

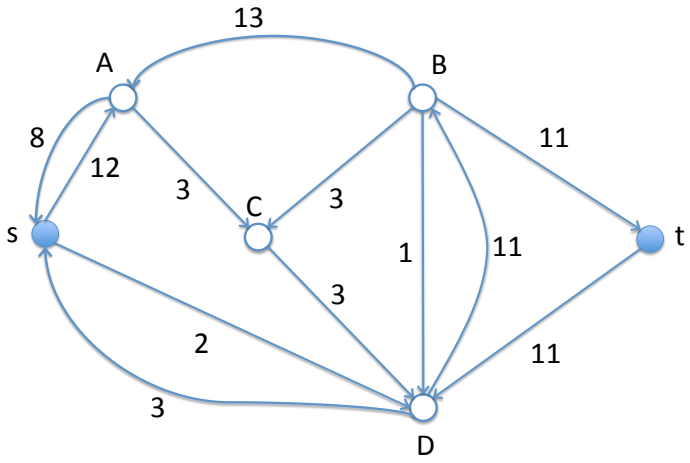
EXAMINER'S USE ONLY			
Question		Mark	Score
1	Maximum Flow, Maximum Fun	16	
2	Linear Programming	20	
3	Hill Climbing	14	
4	Short Proof	10	
5	Hitting Set	20	
6	Sharing the Work	20	
	TOTAL	100	

Problem 1. Maximum Flow, Maximum Fun [16 points]

For both parts of this problem, consider the following flow network, consisting of a source s, a target t, and four intermediate nodes $\{A, B, C, D\}$. Each edge is annoted with a label x/y where x refers to the flow and y refers to the capacity.



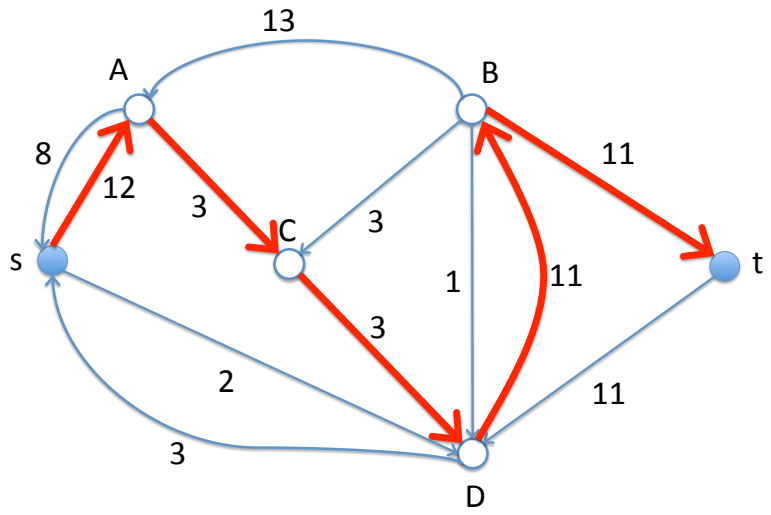
Problem 1.a. [8 points] For the flow network above, draw the residual graph. (Below you will find just the nodes—you need to add all the edges and their residual capacity. You may omit edges with zero residual capacity.)



Solution:
The residual graph contains the edges drawn above. Note that we have omitted edges with zero residual capacity.

Problem 1.b. [8 points] For the flow network depicted (which is identical to the flow network from part (a)), execute one iteration of Ford-Fulkerson, using the *Fattest Path* heuristic to find an augmenting path. Update the flow on the graph. What is the augmenting path? What is the value of the new flow?

Solution:



Fattest augmenting path: s → A → C → D → B → t

Augmenting path = {s,A,C,D,B,t}

New flow value = 14

Above we have drawn the augmenting path on the residual graph. Note that this is the fattest of the two agumenting paths, with botleneck value of 3. The new augmenting path increases the flow by 3, leading to a total flow with value 14.

Problem 2. Linear Programming [20 points]

Problem 2.a. [6 points] Assume you have a linear program P with n variables and m constraints (apart from the natural non-negativity constraints). Assume P is a maximization problem. How many variables and constraints do its dual have?

dual variables = [ ] # dual constraints = [ ]

What is the orientation of P's dual (i.e., maximization, minimization, or other):

Dual orientation = [ ]

Solution: The dual has m variables and n constraints, and is a minimization problem.

Problem 2.b. [4 points] Assume you are given a linear program with three variables x_1, x_2, x_3 that contains the constraint: $x_2 + x_3 = 7$. For example:

$$\begin{array}{lll} \max(x_1 + x_2 + x_3) & \text{subject to:} & \\ & x_1 + x_3 & \leq 10 \\ & x_2 + x_3 & = 7 \\ & x_1, x_2, x_3 & \geq 0 \end{array}$$

How could you write this constraint $x_2 + x_3 = 7$ only in terms of $\leq$ inequalities? That is, how would you write the LP in "normal form" as described in class where all the inequalities (except the non-negativity constraints) are $\leq$?

Solution: The constraint $x_2 + x_3 = 7$ is equivalent to the following two constrints:

$x_2 + x_3 \leq 7$

$x_2 + x_3 \geq 7$

The second of these constraints can be reversed by multiplying by -1, yielding the following two $\leq$ constraints:

$x_2 + x_3 \leq 7$

$-(x_2 + x_3) \leq -7$

Problem 2.c. [10 points] Write down the dual of the following linear program on variables x_1, x_2, x_3, x_4. Specify clearly the variables, the objective function, and the constraints.

$$\begin{array}{ll} \textbf{min}(4x_1 - 9x_2 + 17x_4) & \textbf{subject to:} \\ & 2x_1 + 3x_2 - 5x_3 + x_4 \geq 7 \\ & x_2 \geq 1 \\ & x_1 + x_2 + x_3 - 6x_4 \geq 2 \\ & x_1, x_2, x_3, x_4 \geq 0 \end{array}$$

Solution: The dual will have three variables y_1, y_2, y_3.

$$\begin{array}{ll} \max(7y_1 + y_2 + 2y_3) & \text{subject to:} \\ & 2y_1 + y_3 \leq 4 \\ & 3y_1 + y_2 + y_3 \leq -9 \\ & -5y_1 + y_3 \leq 0 \\ & y_1 - 6y_3 \leq 17 \\ & y_1, y_2, y_3 \geq 0 \end{array}$$

Problem 3. Hill Climbing [14 points]

Problem 3.a. [4 points] Which of the following statements about gradient ascent are always true? Circle the **true** statements and cross out the **false** statements.

Gradient ascent always finds a local minimum (or an ϵ-approximation of a local minimum) if run for long enough.	Gradient ascent always finds a local maximum (or an ϵ-approximation of a local maximum) if run for long enough.
Gradient ascent guarantees that it finds a solution x such that $\|x - x^*\| \leq \epsilon$ when executed for $O(\kappa \log 1/\epsilon)$ iterations on a strongly convex function f with condition number κ where x^* is the input that minimizes f.	A function f is strongly convex (with parameter ℓ) if for any two inputs x and y: $$f(y) \leq f(x) + \nabla f(x) \cdot (y - x) + \frac{\ell}{2}\|\|y - x\|\|^2 \ .$$

Solution: All four statements are false.

a. Gradient ascent may find a point that is neither a local minimum or a local maximum, e.g., a saddle point where $\nabla f = 0$, but that is not a minimum or a maximum.

b. False, as above.

c. Gradient ascent guarantees that after $O(\kappa \log 1/\epsilon)$ time, $|x - x^*| \leq \epsilon |x_0 - x^*|$. However, if x_0 is very far away from x^*, this may not guarantee a very good solution (e.g., if $|x_0 - x^*| = 2^{2^{100/\epsilon}}$).

d. This is the Lipschitz condition, not the condition for strong convexity. It is, in fact, exactly the reverse of the definition of strong convexity.

Problem 3.b. [2 points] Briefly give one reason you might use gradient ascent to find the maximum of a function f (instead of, for example, calculating the maximum in closed form using techniques from calculus).

Solution: Often, it is quite difficult to calculate a closed form solution for a function f, as it may be quite a complicated function. Using gradient ascent allows you to find a maximum even for functions in which computing the closed form solution is infeasible. (This is particularly true if the function is not convex, but instead some very complicated high dimensional surface.)

The process of finding a maximum may also be faster (and easier to automate) using gradient ascent. It can be accomplished numerically (instead of symbolically), for example.

In some cases the function f may not be provided as an equation. Instead, it may be that you can calculate f for given values (e.g., by running a simulation of a physical system), but you are not provided a simple equation for f. In that case, you can still determine the (approximate) gradient by calculating f at nearby points, so you can still use gradient ascent. However, there is no way to solve the problem symbolically.

Problem 3.c. [8 points] Let $f(x, y, z) = x^3 - xy^2 + yz + 4234$ be a function of three variables x, y, and z. Suppose you are using gradient ascent to find a maximum for this function.

Let $(x, y, z) = (10, 20, 30)$. Assume the step size $\alpha = 0.1$. Execute one iteration of gradient ascent. What is the new value of (x, y, z)?

x = ☐ y = ☐ z = ☐

Solution: We first calculate the gradient of f at $(10, 20, 30)$.

$$\begin{aligned} \partial f/\partial x &= 3x^2 - y^2 = 300 - 400 = -100 \\ \partial f/\partial y &= -2xy + z = -400 + 30 = -370 \\ \partial f/\partial z &= y = 20 \end{aligned}$$

So, we conclude that $\alpha\nabla_f(x, y, z) = (-10, -37, 2)$. Thus, $x \leftarrow x + \alpha\nabla_f(x, y, z)$ yields $(x, y, z) = (0, -17, 32)$.

As a sanity check, we observe that $f(10, 20, 30) = 1000 - 4000 + 600 + 4234 = 1834$. By contrast, $f(0, -17, 32) = 0 - 0 - 544 + 4234 = 3690$. Thus, we have found a larger point for the function f. (Notice, of course, that this will not always be true. But for a "smooth" function, if α is sufficiently small, then it will be. So it remains a good sanity check that you did the graident ascent in the direct direction.)

Problem 4. Short proof. [10 points]

Assume you are given a connected, undirected graph $G = (V, E)$ where every node has degree $2k$, for some integer $k \geq 1$. An orientation of G assigns a direction to every edge in E, i.e., for each edge (u, v), it outputs either $(u \to v)$ or $(v \to u)$. The result is a directed graph $orient(G)$.

Prove that there exists an orientation of the graph so that the directed graph $orient(G)$ is strongly connected. (Recall that in a strongly connected graph, there is a directed path between every pair of nodes in both directions.)

Solution: Since graph G is connected and every node has even degree, there exists an Eulerian Cycle that visits every edge in the graph. Orient the edge according to the Eulerian cycle. That is, assume the Eulerian cycle is $e_1, e_2, e_3, \ldots, e_m$. Orient each edge toward the next edge in the cycle, e.g., if $e_i = (u, v)$ and $e_{i+1} = (v, w)$, then orient $e_i = (u \to v)$.

Since the Eulerian cycle visits every node, we know that every pair of nodes are connected by a directed path, i.e., the segment of the Eulerian cycle that connects them (in the correct direction).

Problem 5. Hitting Set [20 points]

Consider the following problem, known as *4-Hitting Set*:

- You are given a set of elements $P = p_1, p_2, \ldots, p_n$.
- You are given a set of sets $S_1, S_2, \ldots, S_m$ where each $S_j \subseteq P$.
- Each set S_j contains at most 4 elements.

The goal is to output the smallest set of elements $H \subseteq P$ that is a *hitting set*, i.e., where each set S_j contains at least one element in P:

$$\forall j : \exists p_i \in H \text{ such that } p_i \in S_j$$

For example, if the sets are $S_1 = \{1, 2, 3\}$, $S_2 = \{2, 5, 7\}$, and $S_3 = \{1, 9, 10\}$, then $H = \{2, 9\}$ is a hitting set: set S_1 is hit by element 2, set S_2 is hit by element 2, and set S_3 is hit by element 9.

Problem 5.a. [8 points] Give an integer linear program that finds the minimum sized hitting set for a given input of elements P and sets $S_1, \ldots, S_m$. (The ILP does not need to be in "normal form".)

Solution: For variables $x_1, \ldots, x_n$:

$$\min \sum_{j=1}^{n} x_j \quad \text{such that } \forall j : \sum_{i:p_i \in S_j} x_i \geq 1$$
$$x_j \in \{0, 1\}$$

Notice that the hitting set problem is essentially identical to the set cover problem, where the sets and elements are reversed. (Notice it is also essentially identical to the problem of vertex cover on a hypergraph.)

Problem 5.b. [12 points] Show how to relax and round the integer linear program from part (a) so as to find a 4-approximation algorithm for the hitting set problem. Prove that your algorithm is correct (i.e., returns a hitting set) and that it is a 4-approximation.

Solution: We relax the ILP by remove the constraint the $x_i \in \{0, 1\}$ and replacing it with a simple non-negativity constraint $x_i \geq 0$.

We round the linear program as follows: If $x_i \geq 1/4$, include it in the output hitting set H. (That is, set $y_i = 1$ if $x_{\geq}1/4$, and otherwise set $y_i = 0$. Include p_i in the output set H if $y_1 = 1$.) Notice that this is a 4-approximation since it increase each x_i by at most a factor of 4:

$$\begin{aligned} OPT &\geq \sum_i x_i \\ &\geq \sum_i y_i/4 \\ &\geq |H|/4 \end{aligned}$$

That is, $|H| \leq 4OPT$ as required.

Next, we show that the resulting set H is a correct hitting set. Observe that for each set $S = \{p_1, p_2, p_3, p_4\}$, we know that $x_1 + x_2 + x_3 + x_4 \geq 1$, by the LP constraint. That means that at least one of these $x_i \geq 1/4$, and so is included in the hitting set. That is, the set S is properly hit.

Problem 6. Sharing the work. [20 points]

Suppose you and your partner are working together on a project. The project consists of n tasks $1, 2, \ldots, n$, and task i takes time t_i to accomplish. Let the total time of all the tasks $M = \sum_{i=1}^{n} t_j$. Assume that no one task is too big, i.e., for all j: $t_j \leq \frac{M}{2}$. You want to divide the tasks into two sets A and B that are as close to the same size as possible. That is, define:

$$\begin{aligned} T_A &= \sum_{j \in A} t_j \\ T_B &= \sum_{j \in B} t_j \end{aligned}$$

You want to minimize $D = |T_A - T_B|$. You decide to divide up the tasks into two sets in the following manner:

GreedyAllocation:

- Set $A = \emptyset$, $B = \emptyset$.
- Let L be the list of n task sizes (in some arbitrary order).
- For $i = 1$ to n do:
  - If $T_A \leq T_B$, then add task $L[i]$ to list A.
  - Otherwise, add task $L[i]$ to list B.

For example, imagine that the tasks sizes $L = \{5, 10, 25, 7, 3, 40\}$. Then, we add the task with size 5 to list A, 10 to list B, 25 to list A, 7 and 3 and 40 to list B. In the end, the total is $T_A = 30$ and $T_B = 75$. Thus, $D = |T_A - T_B| = 45$.

Problem 6.a. [4 points] Give a (simple) example that demonstrates the worst case for this allocation algorithm: give a sequence of tasks where, when allocated in order greedily as above, we find that $T_A = 3T_B$. (Be sure that your tasks obey the rule that for each task j: $t_j \leq M/2$.)

Solution: Consider the tasks with sizes $\{25, 25, 50\}$. The first task is allocated to A, the second task is allocated to B, and the third task is allocated to A, yielding $T_A = 75$ and $T_B = 25$, i.e., $T_A = 3T_B$. Here, $M = 100$, and no task is bigger than 50.

Problem 6.b. [8 points] Assume you run the algorithm, yielding sets A and B with times T_A and T_B. Assume (without loss of generality) that $T_A \geq T_B$. Prove that $T_A \leq 3T_B$.

Hint: Think about the last task that was added to set A. Assume for the sake of contradiction that $T_A > 3M/4$.

Solution: Let t_j be the cost of the last task added to list A. Just prior to adding task t_j, we know that list A had less than list B (otherwise we would have added task j to list B). So we know that $T_B \geq T_A - t_j$.

Assume for the sake of contradiction that $T_A > 3M/4$. Recall that $t_j \leq M/2$. Thus we conclude that $T_B > 3M/4 - M/2 = M/4$. But $T_A + T_B = M$, so if $T_A > 3M/4$, then $T_B \leq M/4$ which is a contradiction.

We conclude that $T_A \leq 3M/4$ and $T_B \geq M/4$, and hence $T_A/T_B \leq 3$, as desired.

Problem 6.c. [8 points] Your partner would like to find a more fair allocation and suggests that if you sort the tasks first, then the greedy allocation algorithm will guarantee that $T_A \leq 2T_B$. Is your partner right? Circle the correct answer:

Yes: Sort from large to small.	Yes: Sort from small to large.	No: Does not help.

Either a give an example that justifies why it does not help, or prove that when the list is sorted properly, then $T_A \leq 2T_B$. As before, assume without loss of generality that $T_A \geq T_B$.

Solution: We show that sorting the tasks from largest to smallest works.

As before, let t_j be the last task added to set A. We already know (see above) that $T_A - T_B \leq t_j$. Our goal is to show that, in this case, $t_j \leq T_B$.

Assume, for the sake of contradiction, that $t_j > T_B$. Since the tasks are added in sorted order from largest to smallest, this implies that all the tasks in B are smaller than t_j, and hence all the tasks in B are added after task t_j. Since list B is empty, this also implies that prior to task j, list A is also empty (otherwise we would have added task j to list B). Thus, task j is the very first task added to the system.

Moreover, all the tasks after task j are added to list B, since j is the last task added to A. From this, we conclude that $T_B = M - t_j \geq M - M/2 \geq M/2$. However, this contradicts our assumption that $t_j > T_B$.

Having shown that $t_j \leq T_B$, we conclude that since $T_A - T_B \leq t_j \leq T_B$, we know that $T_A \leq 2T_B$, as desired.

Scratch Paper

Scratch Paper

Scratch Paper

Scratch Paper

End of Paper