CS570 Spring 2018: Analysis of Algorithms Exam III

	Points		Points
Problem 1	20	Problem 5	15
Problem 2	15	Problem 6	10
Problem 3	15	Problem 7	10
Problem 4	15		
Total: 100 Points			

Instructions:

1. This is a 2-hr exam. Closed book and notes
2. If a description to an algorithm or a proof is required please limit your description or proof to within 150 words, preferably not exceeding the space allotted for that question.
3. No space other than the pages in the exam booklet will be scanned for grading.
4. If you require an additional page for a question, you can use the extra page provided within this booklet. However please indicate clearly that you are continuing the solution on the additional page.
5. Do not detach any sheets from the booklet. Detached sheets will not be scanned.
6. If using a pencil to write the answers, make sure you apply enough pressure so your answers are readable in the scanned copy of your exam.
7. Do not write your answers in cursive scripts.

1) 20 pts
Mark the following statements as **TRUE** or **FALSE**. No need to provide any justification.

[TRUE]
To prove that a problem X is NP-hard, it is sufficient to prove that SAT is polynomial time reducible to X.

[FALSE]
If a problem Y is polynomial time reducible to X, then a problem X is polynomial time reducible to Y.

[TRUE]
Every problem in NP can be solved in polynomial time by a nondeterministic Turing machine.

[TRUE]
Suppose that a divide and conquer algorithm reduces an instance of size n into 4 instances of size $n/5$ and spends $\Theta(\underline{n})$ time in the conquer steps. The algorithm runs in $\Theta(n)$ time.

[FALSE]
A linear program with all integer coefficients and constants must have an integer optimum solution.

[FALSE]
Let M be a spanning tree of a weighted graph G=(V, E). The path in M between any two vertices must be a shortest path in G.

[TRUE]
A linear program can have an infinite number of optimal solutions.

[TRUE]
Suppose that a Las Vegas algorithm has expected running time $\Theta(n)$ on inputs of size n. Then there may still be an input on which it runs in time $\Omega(n^2)$.

[FALSE]
The total amortized cost of a sequence of n operations gives a lower bound on the total actual cost of the sequence.

[FALSE]
The maximum flow problem can be efficiently solved by dynamic programming.

2) 15 pts

Consider a uniform hash function h that evenly distributes n integer keys {0, 1, 2, …, n-1} among m buckets, where $m < n$. Several keys may be mapped into the same bucket. The uniform distribution formally means that Pr($h(x)$=$h(y)$) =1/m for any x, y ∈ {0, 1, 2, …, n-1}. What is the expected number of total collisions? Namely how many distinct keys x and y do we have such that $h(x) = h(y)$?

Solution:

Let the indicator variable X_{ij} be 1 if $h(k_i) = h(k_j)$ and 0 otherwise for all $i \neq j$. Then let X be a random variable representing the total number of collisions. It can be calculated as the sum of all the $X_{ij}'s$: $X = \sum_{i<j} X_{ij}$

Now we take the expectation and use linearity of expectation to get:

$$E[X] = E\left[\sum_{i<j} X_{ij}\right] = \sum_{i<j} E[X_{ij}] = \sum_{i<j} P(h(k_i) = h(k_j)) = \sum_{i<j} \frac{1}{m} = \frac{n(n-1)}{2m}$$

Rubrics:

Describe choosing 2 keys from n keys: 10 pts.

Describe $\sum_{i<j} P\left(h(k_i) = h(k_j)\right)$: 10 pts.

Common mistaks:

1. n/m: The mistake is that it thinks the expected number of collisions for each key is 1/m. 7pts.
2. 2n/m: Similar to 1. 7pts.
3. (n-1)/m: Similar to 1. 7pts.
4. (m+1)/2: Basically, no clue. 3 pts.
5. n(n-1)/m: Duplicates. 13 pts.
6. n^2/m: duplicated. 13 pts
7. (1-(1-1/m)^(n-1))*n: wrong approach. 5 pts.
8. n-m: wrong approach. 3pts.
9. $\frac{1}{m}\sum_{i=0}^{n-1} i$: 15 pts
10. n(n-1)/2: The mistake is that it does not consider collision probability. 10 pts.
11. N(n+1)/2m: minor mistake. 13 pts.

3) 15 pts.

There are 4 production plants for making cars. Each plant works a little differently in terms of labor needed, materials, and pollution produced per car:

	Labor	Materials	Pollution
Plant 1	2	3	15
Plant 2	3	4	10
Plant 3	4	5	9
Plant 4	5	6	7

The goal is to maximize the number of cars produced under the following constrains:

- There are at most 3300 hours of labor
- There are at most 4000 units of material available.
- The level of pollution should not exceed 12000 units.
- Plant 3 must produce at least 400 cars.

Formulate a linear programming problem, using minimal number of variables, to solve the above task of maximizing the number of cars.

Solution:

We need four variables, to formulate the LP problem: x_1, x_2, x_3, x_4, where x_i denotes the number of cars at plant-i.

Maximize $x_1 + x_2 + x_3 + x_4$
s.t.
$x_i \geq 0$ for all i
$x_3 \geq 400$
$2x_1 + 3x_2 + 4x_3 + 5x_4 \leq 3300$
$3x_1 + 4x_2 + 5x_3 + 6x_4 \leq 4000$
$15x_1 + 10x_2 + 9x_3 + 7x_4 \leq 12000$

Rubric:

- Identifying 4 variables (2 pts)
- Correct objective function (3 pts)
- Each condition (2 pts)

4) 15 pts.

Given an undirected connected graph $G = (V, E)$ in which a certain number of tokens $t(v) \geq 1$ placed on each vertex v. You will now play the following game. You pick a vertex u that contains at least two tokens, remove two tokens from u and add one token to any one of adjacent vertices. The objective of the game is to perform a sequence of moves such that you are left with exactly one token in the whole graph. You are not allowed to pick a vertex with 0 or 1 token. Prove that the problem of finding such a sequence of moves is NP-complete by reduction from Hamiltonian Path.

Solution:

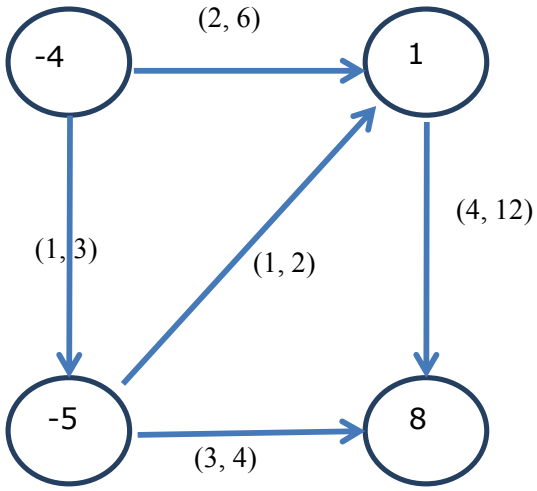
Construction: given a HP in G, we construct G' as follows. Traverse a HP in G and placed 2 tokens on the starting vertex and one token on each other vertex in the path.
Claim: G has a HP iff G' has a winning sequence.
->) by construction before the last move we will end up with a single vertex having two tokens on it. Making the last move, we will have exactly one token on the board.
<-) since there is only one vertex with 2 tokens, we will start right there playing the game. Each next move is forced. When we finish the game, we get a sequence moves which represents a HP.

Rubrics:

- Didn't prove it is in NP: -5
- Didn't prove it is NP-hard: -10
- Assigned two tokens on one random vertex instead of the starting vertex of Hamiltonian path: -2 (Since it is a reduction from Hamiltonian path, not from Hamiltonian cycle, 2 tokens should be assigned on the starting vertex)
- Assigned wrong number of tokens on one vertex: -3
- Assigned wrong number of tokens on two vertices: -6
- Assigned wrong number of tokens on three or more vertices (considered as not a valid reduction from Hamiltonian path): -7
- Not a valid reduction from Hamiltonian path: -7

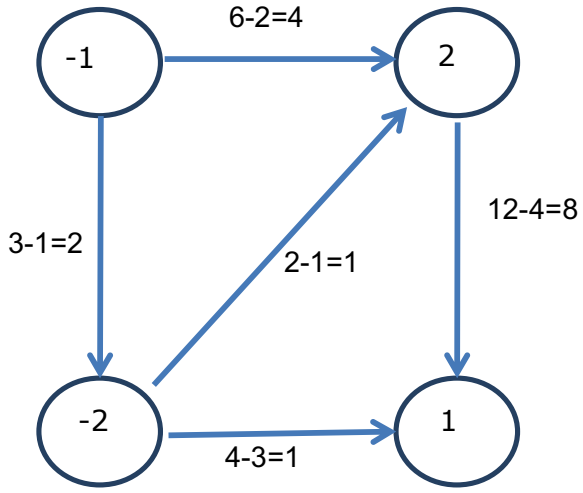
5) 15 pts.

In the network below, the demand values are shown on vertices (supply value if negative). Lower bounds on flow and edge capacities are shown as (lower bound, capacity) for each edge. Determine if there is a feasible circulation in this graph. You need to show all your steps.



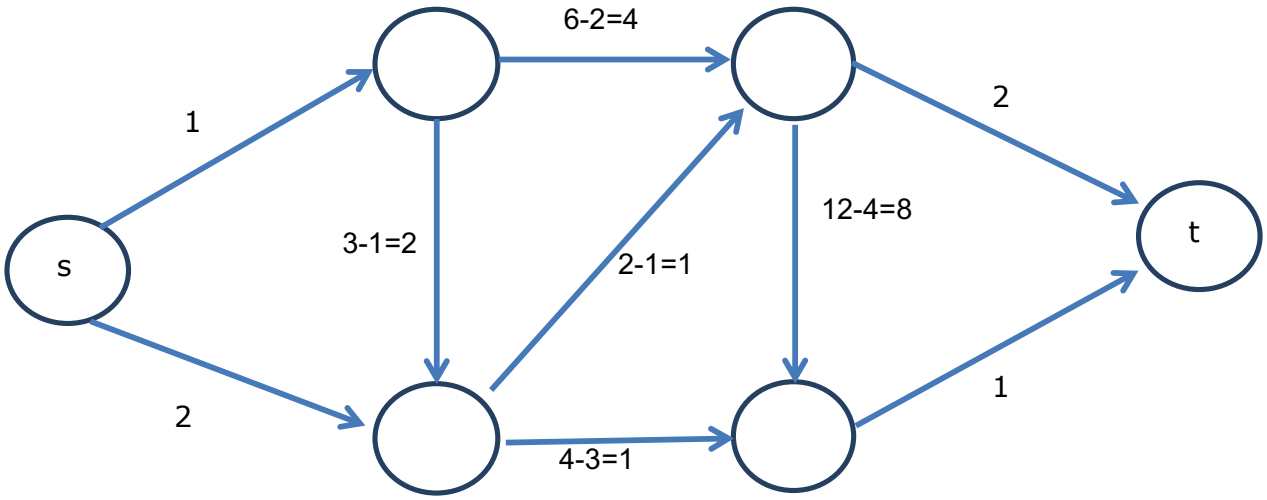
a) Turn the circulation with lower bounds problem into a circulation problem without lower bounds (6 pts)

- **Each wrong number for nodes: -1**
- **Each wrong number for edges: -0.5**



b) Turn the circulation with demands problem into the max-flow problem (5 pts)

- **s and t nodes on right places: each has 0.5 point**
- **directions of edges from s and t: each direction 0.5 point**
- **values of edges from s and t: each value 0.5 point**

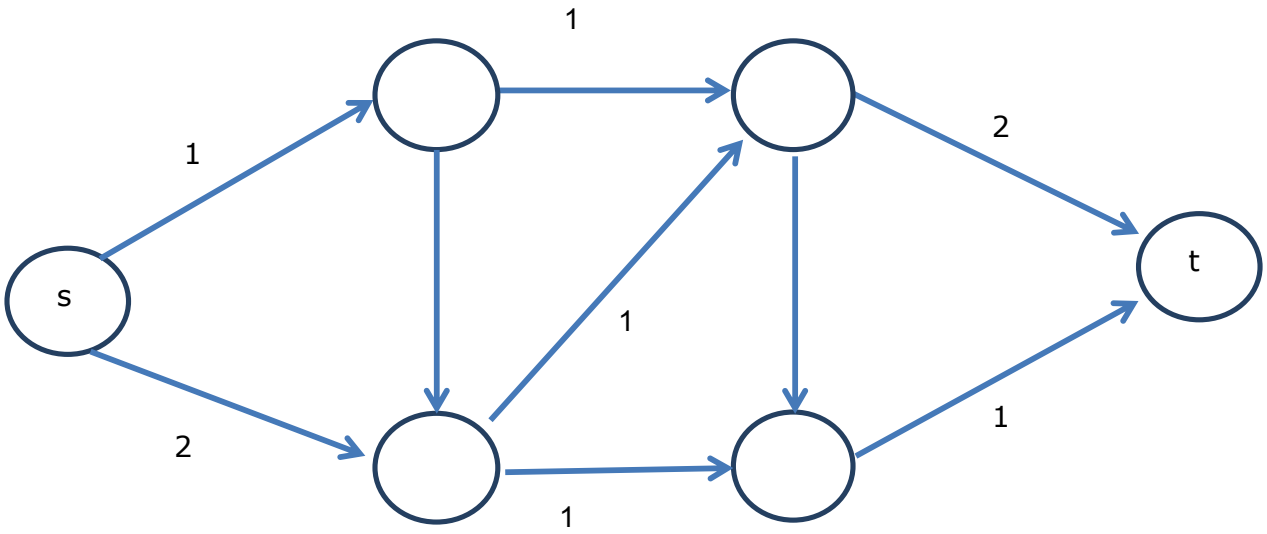


c) Does a feasible circulation exist? Explain your answer. (4 pts)

Solution.

Yes. There is a s-t flow |f|= 2+1=1+2=3. It follows, there is a feasible circulation.

- **Explanation: 2 points.**
- **Correct Graph/flow: 2 points.**



6) 10 pts.

We wish to determine the most efficient way to deliver power to a network of cities. Initially, we have n cities that are all connected. It costs $c_i \geq 0$ to open up a power plant at city i. It costs $r_{ij} \geq 0$ to build a cable between cities i and j. A city is said to have power if either it has a power plant, or it is connected by a series of cables to some other city with a power plant (in other words, the cables act as undirected edges). Devise an efficient algorithm for finding the *minimum cost* to power all the cities.

Solution

Consider a graph with the cities at the vertices and with weight r_{ij} on the (undirected) edge between city i and city j. Add a new vertex s and add an edge with weight ci from s to the ith city, for each i. The minimum spanning tree on this graph gives the best solution (power stations need to be opened at those cities *i* for which (s,i) is part of the MST.)

Common Mistakes:

- Dynamic Programming: There does not exist a predefined order in which the nodes/edges will be processed and because of this you cannot define the problem based on smaller sub-problems and hence, none of the solutions which were based on Dynamic Programming were correct.
- Max-Flow/Min Cut: It was a surprise to see many ran a Max Flow algorithm to minimize the overall cost of edges. The minimum cut in a network has nothing to do with minimizing flow!! Furthermore, in this problem we want connectivity in the graph and edges on the min cut don't guarantee any level of connectivity.
- ILP: I can't recall anybody correctly formalizing the problem as in ILP. However, even a correct formulization is going to be NP-Hard and far from efficient. A maximum of 3 points was given to a CORRECT ILP solution.
- Dijkstra's algorithm is for finding the shortest path between a single point and every other node in the graph. The resulting Tree from running Dijkstra, is not necessarily an MST so it's not the correct approach to solve this problem.

7) 10 pts.

We want to break up the graph $G = (V, E)$ into two disjoint sets of vertices $V=A \cap B$, such that the number of edges between two partitions is as large as possible. This problem is called a max-cut problem. Consider the following algorithm:

- Start with an arbitrary cut C.
- While there exists a vertex v such that moving v from A to B increases the number of edges crossing cut C, move v to B and update C.

Prove that the algorithm is a 2-approximation.
Hint: after algorithm termination the number of edges in each of two partitions A and B cannot exceed the number of edges on the cut C.

Solution

Let w(u,v) = 1, there exists an edge between u and v, and 0 otherwise. Let

$$W = \sum_{(u,v)\in E} w(u,v)$$

By construction, for any node u∈A:

$$\sum_{v\in A} w(u,v) \le \sum_{v\in B} w(u,v)$$

Here, the left hand side is all edges in partition A. The RHS – the crossing edges. If we sum up the above inequality for all u∈A, the LHS will contain the twice number of edges in A.

$$2\sum_{(u,v)\in A} w(u,v) \le \sum_{u\in A, v\in B} w(u,v) = C$$

We do the same for any node in B:

$$2\sum_{(u,v)\in B} w(u,v) \le \sum_{u\in A, v\in B} w(u,v) = C$$

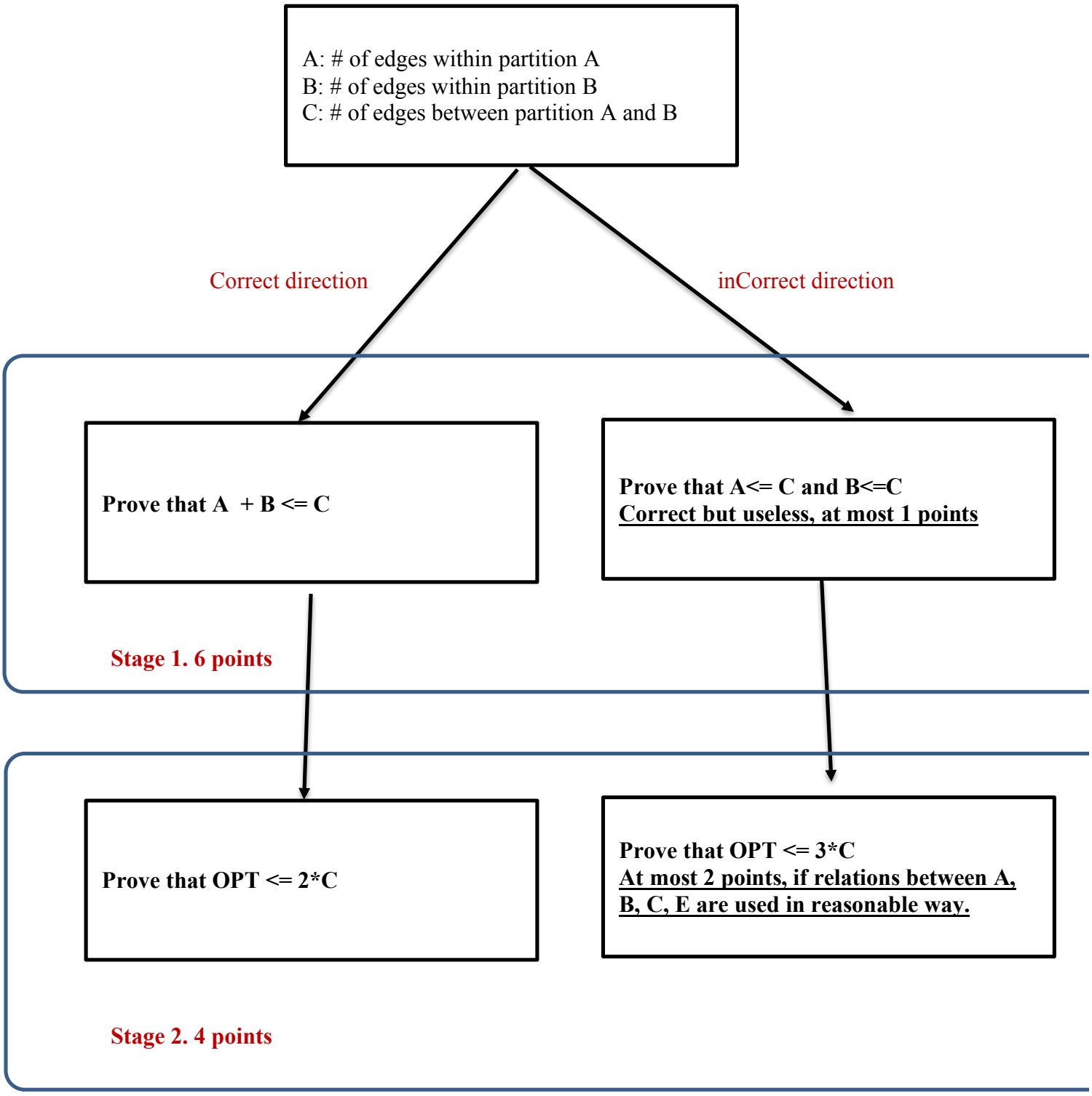
Add them up

$$\sum_{(u,v)\in A} w(u,v) + \sum_{(u,v)\in B} w(u,v) \le C$$

Next we add edges on the cut C to both sides.

$$\sum_{(u,v)\in A} w(u,v) + \sum_{(u,v)\in B} w(u,v) + \sum_{u\in A, v\in B} w(u,v) = W \le 2C$$

Clearly the optimal solution OPT≤W, thus OPT≤W ≤2C. It follows, OPT/C ≤ 2.


A: # of edges within partition A
B: # of edges within partition B
C: # of edges between partition A and B
Correct direction
inCorrect direction
Prove that A + B <= C
Prove that A<= C and B<=C
Correct but useless, at most 1 points
Stage 1. 6 points
Prove that OPT <= 2*C
Prove that OPT <= 3*C
At most 2 points, if relations between A, B, C, E are used in reasonable way.
Stage 2. 4 points

1. The solution can be divided into two parts. **Part 1** is worth 6 points, and **part 2** is worth 4 points.
    1. Proving the hint, i.e. "numbers of edges in partition A and partition B cannot exceed the number of edges on the cut C". It is most important and challenging part of this problem.
    2. Prove the algorithm is a 1/2-approximation.
2. About part 1, here are two acceptable solutions and some solutions not acceptable:
    1. If your solution is same to the official solution, using inequalities to rigorously prove the hint, it is perfect, 6 points.
    2. Here is a weaker version. A the end of algorithm, for **any** node **u** in the graph, without loss of generality assume it is in part A, **the number of edges** connecting it to nodes in part B (i.e. edges in the cut) is greater than or equal to **the number of edges** connecting it to other nodes in part A. So, the **total number of edges** in the cut is greater than or equal to the **total number of edges** within each part. This solution did not consider the important fact that both inter-partition and intra-partition edges are counted twice. 3 ~5 points according to the quality of statements.
    3. One incorrect proof: some answers declare that C is increasing and A + B is decreasing when calling steps 2 of the algorithm. This is a correct but useless statement. 0 points
    4. YOU MUST PROVE THE HINT. SIMPLY STATING OR REPHRASING IT GET 0 POINTS IN THIS STEP. Here is one example that cannot get credits: "According to the algorithm, we move one vertex from A to B if it can increase the number of edges in cut C. Then the number of edges in C is more than edges in partition A and partition B." 0 points
    5. "If move v from A to B can increase the number of edges cross cut C, it means v connect to more vertex in A than in B". It is rephrasing the algorithm, and failed to get the key point. If the above statement is changed to "If move v from A to B can increase the number of edges cross cut C, it means v connect to more vertex in different partition than in the same partition", it will get at least 3 points.
3. About part 2, the correct answer should use relation between OPT, E, A, B, C. Missing key inequalities like OPT <= E, will lead to loss of grades.
4. If the answer is incorrect, you may get some points from some steps that make sense. If the answer is on the incorrect direction in the above diagram:
    1. Many students attempted to PROVE the edges_in_partition_A <= C, and edges_in_partition_B <= C, but not proving edges_in_partition_A + edges_in_partition_B <= C. This is trivial according to step 2 of algorithm, and is not useful for proving the final statement. This will be treated as "correct but useless/irrelevant statements". AT MOST 1 point.
    2. From above inequalities, OPT<=3*C is a correct conclusion. It is different from the question, but since the logics in this stage is same, you can get at most 2 points out of 4.