

MATH 3411 (Ng/Fall 2011)
Review for Final Examination
Final is Comprehensive, and it is scheduled on
Thursday December 15 at 4:00pm-6:00pm
for class on December 8, 2011.

1. **Reviews for Exams 1,2, & 3**

2. **Applications**

Given a description of a real-world problem, you should know how to formulate it as a combinatorial optimization problem, and know which correct optimization problem to apply in order to solve the real-world problem.

3. **Shortest Directed Path (SP)**

- (a) You should know statement of the problem for an (SP).
- (b) You should know how to model an application problem as one of shortest path, i.e. know what is given and what you want to find.
- (c) You should know the required restrictions for the *Dijkstra* and the *Floyd-Warshall's* algorithm to work.
- (d) You should know how to apply the *Dijkstra* and the *Floyd-Warshall's* algorithm to find the shortest directed path(s). At the end of the algorithm, you should know how to identify the optimal solutions (shortest directed paths) and the optimal value (the length of the shortest directed paths.)
 (Note: there could be multiple optimal solutions, but there can only be a **unique** optimal value, if one exists.)
- (e) You should know how to apply $F - W$'s algorithm to finding a negative weight directed cycle in a directed graph, if one exists.
- (f) You should know the rationale behind why *Dijkstra* and the *Floyd-Warshall's* algorithms work.

4. **Maximum $s-t$ Flow Problem (MF) and Minimum Capacity $s-t$ Cut Problem (MC)**

- (a) You should know statement of the problem for (MF).
- (b) You should know statement of the problem for (MC).
- (c) You should know how to model an application problem as one of (MF) or (MC), i.e. know what is given and what you want to find.
- (d) You should know how to use the Ford-Fulkerson's (FF) algorithm to find a maximum feasible $s - t$ flow.
 There is a big difference between a maximum feasible $s - t$ flow and the *value* of a maximum $s - t$ flow.
- (e) Be familiar with all the weak duality type lemma, the strong duality type lemma, and the max-flow min-cut theorem.
- (f) You should be familiar with all other results relating the two different combinatorial optimization problems, (MF) and (MC), in particular, how (MC) is solved once (MF) is solved.
- (g) **NEVER** solve (MC) via trials of several sets, S , or by total enumeration.
 Reason: your final exam is NOT short, and you need to optimize your time.

(h) You should be familiar with the terms:

- i. an *s-t flow*, i.e., an s-t flow is an assignment of values on the **arcs** of the given directed graph,
- ii. a *feasible s-t flow*, i.e. a feasible s-t flow is an s-t flow that satisfies two conditions, namely, the flow on each arc must be non-negative and cannot exceed its capacity, and, total flow into any vertex $k \in V \setminus \{s, t\}$ must be the same as the total flow out of vertex k .
- iii. the *value* of a feasible s-t flow, i.e. the value of a feasible s-t flow is the net flow out of the source, s , or the net flow into the sink, t , or the net flow across the arcs of any $s - t$ cut.
- iv. Given a set of vertices S such that $s \in S$ and $t \notin S$, an *s-t cut defined by S* denoted $\delta(S)$, is a set of **arcs** (not vertices) that go from vertices in S to vertices NOT in S OR from vertices not in S to vertices in S .

Technically,

$$\delta(S) = \{(i, j) \in A : (i \in S \wedge j \notin S) \vee (i \notin S \wedge j \in S)\}$$

- v. Given an $s - t$ cut, the *capacity* of the $s - t$ cut, denoted $cap(\delta(S))$ is the sum of all the capacities of the *forward arcs* in $\delta(S)$. (Forward arcs in an s-t cut are arcs (i, j) in the $s - t$ cut such that $i \in S \wedge j \notin S$.)

For instance, from the directed graph $G = (V, A)$ in **Figure 1**, with capacities $u_{ij} \geq 0$, and with $s = 1$ and $t = 2$, the following are examples of $s - t$ cuts defined by vertex sets \tilde{S} .

- Let $\tilde{S} = \{1, 6\}$. Then, the $s - t$ cut defined by \tilde{S} , $\delta(\tilde{S})$ and its capacity, $cap(\delta(\tilde{S}))$ are, respectively,

$$\delta(\tilde{S}) = \{(1, 3), (1, 4), (5, 1), (1, 5), (4, 6), (6, 5), (6, 4), (6, 2)\}$$

and

$$cap(\delta(\tilde{S})) = u_{13} + u_{14} + u_{15} + u_{65} + u_{64} + u_{62} = 6 + 4 + 9 + 5 + 16 + 11 = 51$$

- Let $\tilde{S} = \{1, 3, 4\}$. Then, the $s - t$ cut defined by \tilde{S} , $\delta(\tilde{S})$ and its capacity, $cap(\delta(\tilde{S}))$ are, respectively,

$$\delta(\tilde{S}) = \{(4, 2), (6, 4), (5, 4), (4, 6), (1, 5), (5, 1), (1, 6)\}$$

and

$$cap(\delta(\tilde{S})) = u_{42} + u_{46} + u_{15} + u_{16} = 7 + 11 + 9 + 10 = 37$$

Note: DO NOT FORGET THAT A CUT INCLUDES ALL FORWARD AND REVERSE ARCS

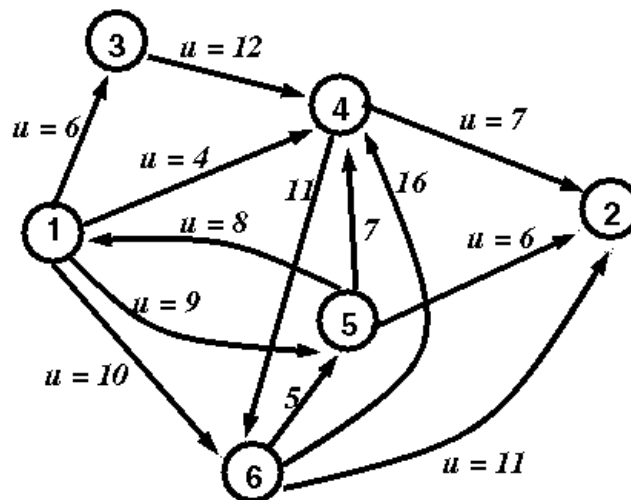


Figure 1 : Graph for examples of $s - t$ Cuts

5. TSP and Vertex coloring Problems

- Given a graph $G = (V, E)$, you should know what a *Hamiltonian cycle* of G is, i.e. a cycle of G that goes through every vertex in V .
- Given a graph $G = (V, E)$ and *weights*, w_{ij} for each $(i, j) \in E$, you should know what the *Travelling Salesman Problem* (TSP) is.
- Given a graph $G = (V, E)$, you should know what a *vertex coloring* of G is.
- Given a graph $G = (V, E)$, you should know what is the *chromatic number* of G , $\chi(G)$, is.
- You should know what the chromatic numbers are for certain classes of graphs such as *cycles, bipartite graphs, wheels, planar graphs, etc...*
- You should know how to apply vertex-coloring of a graph to scheduling problems.
- Nice** versus **Hard** combinatorial optimization problems.
 - You should know that combinatorial optimization problems such as the *Minimum Cost Spanning Tree Problem*, the *Postman Problem*, the *Shortest di-Path* problem with no negative weight directed cycles, and the *Maximum $s-t$ Flow* or the *Minimum $s-t$ cut* problems are considered *nice* combinatorial optimization problems because there are efficient algorithms to solve them. By efficient, we meant that no matter how large the input information is, the known algorithms to solve these problems will take, in the **worst case scenario**, an amount of steps that is directly proportional to a **polynomial function** of the size of the input, for instance, $|V|$ or $|E|$.
 - On the other hand, combinatorial optimization problems such as the *Travelling Saleman Problem* and the *Minimum Vertex Coloring Problem* are both *NP - Hard* optimization problems, meaning that to this date, there is **NO** known efficient-time algorithms to solve these problems to optimality, and most likely, there probably will not be an efficient way to solve these problems.

Hence, problems such as the (TSP) and finding $\chi(G)$ are the types of mathematical optimization problems that are just plain **cool!** Trying to find **good heuristics** (approximation methods) for solving **Hard** problems such as the (TSP) and $\chi(G)$, or finding classes of graphs where (TSP) and $\chi(G)$ can be solved efficiently, are the problems that make my life interesting. 😊