ADVALGO QUICK EXAM SUMMARY

GABRIEL ROVESTI

Disclaimer

The following file is made to have all of the exercises of the exam solved and each category present here.

1 SUMMARY

2 EXAM – FIRST PART

2.1 COMPLEXITIES AND NP-HARD PROBLEMS

2.1.1 Algorithms

- DFS/BFS = $O(n + m)$
  - Also called in exams "Graph connectivity/Connected components"
- Minimum Spanning Tree (MST)
  - Prim's algorithm = $O(m * n)$
    - Prim with heaps = $O(m \log(n))$
  - Kruskal's algorithm = $O(m * n)$
    - Kruskal with Union-Find = $O(m \log(n))$ = best algorithm
- Single-source shortest paths (SSSP)
  - Dijkstra's algorithm = $O(m * n)$
    - Dijkstra with heaps= $O(m * n(\log(n))$ = best algorithm
  - Bellman-Ford's algorithm = $O(m * n)$
- All-pairs shortest paths (APSP)
  - Bellman-Ford with dynamic programming = $O(n^3 \log(n))$
  - Floyd-Warshall = $O(n^3)$
- Maximum flow
  - Ford-Fulkerson = $O(m|f^*|)$

2.1.2 NP-Hard Problems

- NP-Hard problems (seen in the course)
  - TSP – Traveling Salesperson Problem
  - Metric TSP
  - Maximum Independent Set (or Maximum independent set)
  - Vertex cover (or Minimum Vertex Cover)
  - 3SAT
  - Hamiltonian circuit
  - Clique (or Maximum Clique)
  - Set Cover

2.2 GRAPHS EXERCISES

2.2.1 Bellman-Ford

Question 1 (4 points) Consider the following directed, weighted graph, represented by an adjacency matrix where each numerical value represents the weight of the corresponding edge, and where the symbol '–' indicates the absence of the edge between the corresponding vertices.

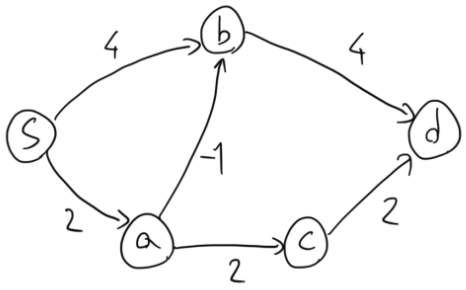
	s	a	b	c	d
s	-	2	4	-	-
a	-	-	-1	2	-
b	-	-	-	-	4
c	-	-	-	-	2
d	-	-	-	-	-

(a) Draw the graph.

(b) Run the Bellman-Ford algorithm on this graph, using vertex s as the source. You are to return the trace of the execution, i.e. a table with rows indexed by vertices and columns indexed by iteration indexes (starting from 0) where each entry contains the estimated distance between s and that vertex at that iteration.

Solution:

(a)



(b)

	0	1	2	3	4
s	0	0	0	0	0
a	∞	2	2	2	2
b	∞	4	1	1	1
c	∞	∞	4	4	4
d	∞	∞	8	5	5

2.2.2 Dijkstra

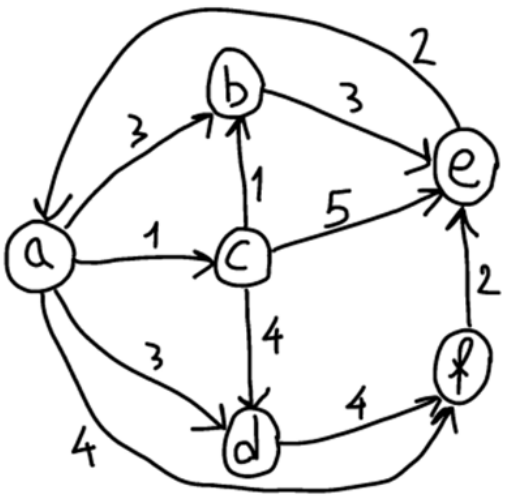
Question 2 (4 points) Consider the following directed and weighted graph, represented by an adjacency matrix where each numerical value represents the weight of the corresponding arc and where the symbol '–' indicates the absence of the arc between the corresponding vertices.

	a	b	c	d	e	f
a	-	3	1	3	-	4
b	-	-	-	-	3	-
c	-	1	-	4	5	-
d	-	-	-	-	-	4
e	2	-	-	-	-	-
f	-	-	-	-	2	-

(a) Draw the graph.

(b) List the lengths of the shortest paths from vertex a to all the other vertices of the graph in the order they are determined by Dijkstra's algorithm.

1.



2. $a-c:1$, $a-b:2$, $a-d:3$, $a-f:4$, $a-e:5$.

2.2.3 Kruskal and Prim

Question 1 (6 points) Consider the following weighted graph, represented by an adjacency matrix where each numerical value represents the weight of the corresponding edge, and where the symbol '–' indicates the absence of the edge between the corresponding nodes.

	a	b	c	d	e	f
a	-	3	7	-	1	5
b		-	6	8	-	-
c			-	2	9	-
d				-	-	-
e					-	4
f						-

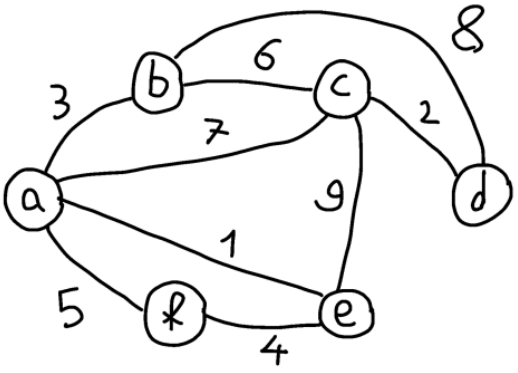
(a) Draw the graph.

(b) List the edges of the minimum spanning tree in the order they are selected by Kruskal's algorithm.

(c) List the edges of the minimum spanning tree in the order they are selected by Prim's algorithm starting at node c.

Solution:

(a)



(b) $(a,e),(c,d),(a,b),(e,f),(b,c)$.

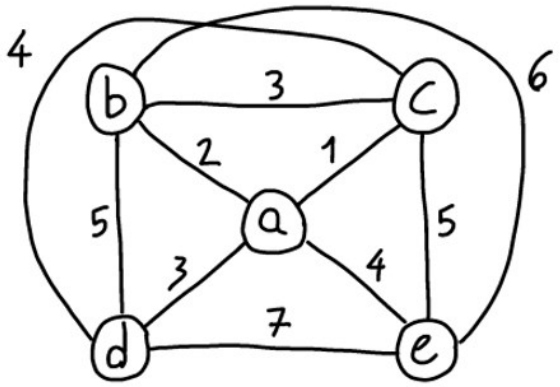
(c) $(c,d),(b,c),(a,b),(a,e),(e,f)$.

Domanda 2 (6 punti) Si consideri il seguente grafo completo pesato, rappresentato tramite una matrice di adiacenza dove ogni valore numerico rappresenta il peso del lato corrispondente.

	a	b	c	d	e
a	-	2	1	3	4
b		-	3	5	6
c			-	4	5
d				-	7
e					-

1. Disegnare il grafo.
2. Elencare i lati del minimum spanning tree nell'ordine in cui sono selezionati dall'algoritmo di Prim a partire dal nodo a.
3. Siccome i pesi dei lati soddisfano la disuguaglianza triangolare, si può applicare l'algoritmo di 2-approssimazione APPROX_T_TSP visto in classe per il TSP. Considerando i nodi ordinati secondo l'ordine alfabetico, fornire l'output dell'algoritmo APPROX_T_TSP quando eseguito su questo grafo.

Soluzione:

1. 

2. $(a, c), (a, b), (a, d), (a, e)$.
3. Considerando i nodi ordinati secondo l'ordine alfabetico, l'algoritmo APPROX_T_TSP esegue l'algoritmo di Prim a partire dal nodo a. Un possibile output è quindi a, c, b, d, e, a.

3 EXAM – SECOND PART

3.1 HOW TO DO REDUCTIONS

$$Y \leq_p X$$

Here's a general structure you can follow when solving a reduction problem to prove that a problem X is NP-hard by reducing a known NP-hard problem Y to X:

1. *Introduction*

- Goal: To prove that problem X is NP-hard by reducing a known NP-hard problem Y to X.

2. *Problem Definition*

- Define problem Y (the known NP-hard problem)
    - Specify the input format and the desired output
- Define problem X (the problem you want to prove is NP-hard)
    - Specify the input format and the desired output

3. *Reduction process*

- Given an instance of problem Y, construct an instance of problem X
- Specify the steps to transform an instance of Y into an instance of X
- Explain how the input of Y is used to create the input of X
- Define any additional variables or structures needed for the reduction

4. *Correctness proof*

- If there is a solution to the instance of Y
    - then there is a corresponding solution to the constructed instance of X
- If there is a solution to the constructed instance of X
    - then there is a corresponding solution to the instance of Y

5. *Polynomial-time reduction*

- Argue that the reduction can be performed in polynomial time, in terms of size and time

3.1.1 Complete Example

$$Ham \leq_p TSP$$

Input:

- For Ham, the input is a graph $G = (V, E)$, where V is the set of vertices and E is the set of edges.
- For TSP, the input is a complete weighted graph $G' = (V', E', w)$, where V' is the set of vertices, E' is the set of edges, and w is a weight function assigning a non-negative weight to each edge.

Output:

- For Ham, the output is "Yes" if the graph G contains a Hamiltonian cycle (a cycle that visits each vertex exactly once) and "No" otherwise.
- For TSP, the output is the minimum total weight of a Hamiltonian cycle in the graph G'.

Reduction:

Given an instance of Ham (a graph G), we construct an instance of TSP (a complete weighted graph G') as follows:

1. Set $V' = V$, so the vertices of G' are the same as the vertices of G.

2. For each edge $(u, v) \in E$, set the weight $w(u, v) = 1$ in G'.

3. For each pair of vertices $(u, v) \notin E$, set the weight $w(u, v) = 2$ in G'.

Now, we have an instance of TSP (the complete weighted graph G'). We claim that G has a Hamiltonian cycle if and only if the minimum total weight of a Hamiltonian cycle in G' is exactly $|V|$.

To show the reduction is correct, we prove the following:

1. If G has a Hamiltonian cycle, then there exists a Hamiltonian cycle in G' with a total weight of |V|.

- Suppose G has a Hamiltonian cycle. This means there is a cycle that visits each vertex exactly once using only the edges in E.
- In the constructed graph G', the edges from the Hamiltonian cycle in G have a weight of 1, and there are $|V|$ such edges.
- Thus, the total weight of this Hamiltonian cycle in G' is exactly $|V|$.

2. If there exists a Hamiltonian cycle in G' with a total weight of $|V|$, then G has a Hamiltonian cycle.

- Suppose there is a Hamiltonian cycle in G' with a total weight of $|V|$.
- Since the minimum weight of any edge in G' is 1, the Hamiltonian cycle in G' must use only edges with weight 1.
- By construction, the edges with weight 1 in G' correspond to the edges in G.
- Therefore, the Hamiltonian cycle in G' corresponds to a Hamiltonian cycle in G.

This reduction shows that if we can solve TSP efficiently, we can also solve Ham efficiently. We construct an instance of TSP from an instance of Ham, solve TSP, and then interpret the solution to determine if the original graph G has a Hamiltonian cycle.

3.1.2 Known reductions seen inside the course

These are collected here just to clearly see them:

- $3SAT \leq_p IndependentSet$
- $Clique \leq_p Independent\ Set$
- $Vertex\ Cover \leq_p Independent\ Set$
- $Ham \leq_p TSP$
- $TSP \leq Metric\ TSP$
- $Vertex\ Cover \leq_p Set\ Cover$

3.2 HOW TO DO APPROXIMATION ALGORITHMS

Definition: Let Π be an optimization problem and let A_Π be an algorithm for Π that returns, $\forall i \in I, A_\Pi(i) \in S_i$ (in other words, the choice the algorithm makes). We say that A_Π has an approximation factor of $\rho(n)$ if $\forall i \in I$ such that $|i| = n$ we have (for each one, the concrete translation in problems):

1. minimization problem (basically, an explicit *lower-bound* of the optimal solution)

$$\frac{c\big(A_\Pi(i)\big)}{c\big(s^*(i)\big)} \leq \rho(n)$$

$$\frac{Greedy}{OPT} \leq \rho(n)$$

Here I call V the choice of the algorithm, say it's Vertex Cover.

So, in a logic of a X-approximation algorithm $\frac{|V'|}{|V^*|} \leq X$:

a) Upper bound to the cost of V' (which is our solution, greedy choice made by us)

- The upper bound means instances for which the algorithm does not improve
- It is the worst-case performance of the algorithm compared to the optimal solution

b) Lower bound to the cost of V^* (which is the optimal solution, selected by the algorithm)

- The lower bound means a value the algorithm is guaranteed to be greater or equal to, hence the smallest possible cost case (base case)
- Represents the best-case performance of the algorithm compared to the optimal solution

2. maximization problem (basically, an explicit *upper-bound* of the optimal solution)

$$\frac{c\big(s^*(i)\big)}{c\big(A_\Pi(i)\big)} \leq \rho(n)$$

$$\frac{OPT}{Greedy} \leq \rho(n)$$

So, in a logic of a X-approximation algorithm $\frac{|V^*|}{|V'|} \leq X$:

a) Upper bound to the cost of V^* (which is the optimal solution, selected by the algorithm)

- Same observations as before

b) Lower bound to the cost of V' (which is our heuristic solution, greedy choice made by us)

- Same observations as before

Most common case is the *2-approximation algorithm*. What does this even mean? It's an algorithm which returns a solution whose cost is at most *twice* the optimal. Specifically:

- it gives solutions that never cost more than twice that of optimal if it is a minimization problem
- or never provide less than half the optimal value if it is a maximization problem

So, it's something in line of (given the structure above):

$$|V'| \leq choice \leq |V^*| \; for \; minimization \; problems$$

$$|V^*| \leq choice \leq |V'| \; for \; maximization \; problems$$

Sometimes, you will be asked to show an approximation is *tight*:

- that depends on your definition of approximation ratio
- normally the approximation ratio is defined as the worst ratio between optimal solution and the one produced by your algorithm
- if this is the case, all you need to show that the ratio is tight is come up with one bad example, which shows it works for all sizes

Specifically, if you have for instance: show the ratio is tight for a 2-approx algorithm, it means, taking for instance Vertex Cover that the ratio is <u>exactly</u> 2.

$$\frac{|V'|}{|V^*|} = 2$$

3.2.1 Known approximation algorithms seen in course

These are collected here just to clearly see them (note: they are <u>all</u> minimization problems):

- 2-approximation algorithm for Vertex Cover
  - $\frac{|V'|}{|V^*|} \leq 2$
    - $|V'| \leq choice \leq 2|V^*|$
- 2-approximation algorithm for Metric TSP (where $H = tour$)
  - $\frac{w|H|}{w|H^*|} \leq 2$
    - $w(H) \leq choice \leq 2w(H^*)$
- 1.5 approximation algorithm for Metric TSP
  - $\frac{|H^*|}{|H'|} \leq 1.5 = \frac{3}{2}$
    - $w(H) \leq choice \leq \frac{3}{2}w(H^*)$
- logarithmic algorithm ($\log_2(n) + 1$) for Set Cover
  - $\frac{|S^*|}{|S'|} \leq \log_2(n) + 1$

3.3 CHERNOFF BOUNDS AND HIGH PROBABILITY

Consider the following footprint exercise – as he will say multiple times, he will give you the specific Chernoff bound:

> Let $X_1, X_2, \ldots, X_n$ be independent indicator random variables such that $\Pr(X_i = 1) = 1/(4e)$. Let $X = \sum_{i=1}^{n} X_i$ and $\mu = E[X]$. By applying the following Chernoff bound, which holds for every $\delta > 0$,
>
> $$\Pr(X > (1+\delta)\mu) < \left(\frac{e^{\delta}}{(1+\delta)^{1+\delta}}\right)^{\mu}$$
>
> prove that
>
> $$\Pr(X > n/2) < \frac{1}{(\sqrt{2})^n}.$$

To apply the Chernoff bound, we get the value which has to be greater from – the bound – as the $(1+\delta)\mu$, since this is the bound.

Then, we apply to each variable the Chernoff bound, so to have the expected value be the same probability applied n times, so $\Pr(X_i = 1) = \frac{1}{4e}$ becomes $\sum_{i=1}^{n} E[X_i] = \frac{n}{4e}$.

We have to set up the target bound, and now you will see precisely why δ gets that value:

2. Set up the target bound: We want to find δ such that:

$$\Pr\left(X > \frac{n}{2}\right)$$

matches the Chernoff bound. So, we equate:

$$\frac{n}{2} = (1+\delta)\mu$$

Substituting $\mu = \frac{n}{4e}$:

$$\frac{n}{2} = (1+\delta)\cdot\frac{n}{4e}$$

3. Solve for δ:

$$\frac{n}{2} = (1+\delta)\cdot\frac{n}{4e}$$

Dividing both sides by $\frac{n}{4e}$:

$$\frac{2e}{1} = 1+\delta$$

Therefore:

$$\delta = 2e - 1$$

Then, see all of these passages:

Substituting $\delta = 2e-1$ and $\mu = \frac{n}{4e}$:

$$\Pr\left(X > (1+2e-1)\,\mu\right) < \left(\frac{e^{2e-1}}{(2e)^{2e}}\right)^{\frac{n}{4e}}$$

Using the Chernoff bound formula:

$$\Pr\left(X > (1+\delta)\mu\right) < \left(\frac{e^{\delta}}{(1+\delta)^{1+\delta}}\right)^{\mu}$$

5. Simplify the expression:

$$\Pr\left(X > \frac{n}{2}\right) < \left(\frac{e^{2e-1}}{(2e)^{2e}}\right)^{\frac{n}{4e}}$$

$$= \left(\frac{e^{2e-1}}{(2e)^{2e}}\right)^{\frac{n}{4e}}$$

$$= \left(\frac{e^{2e-1}}{(2e)^{2e}}\right)^{\frac{n}{4e}}$$

6. Further simplify: Recognize that $\frac{e^{2e-1}}{(2e)^{2e}} = \left(\frac{e}{2e}\right)^{2e} = \left(\frac{1}{2}\right)^{2e} = \frac{1}{(2e)^{2e}}$:

$$\left(\frac{e^{2e-1}}{(2e)^{2e}}\right)^{\frac{n}{4e}} = \left(\frac{1}{2e}\right)^{\frac{n}{4}}$$

Simplify further:

$$\left(\frac{1}{2e}\right)^{\frac{n}{4}} = \left(\frac{1}{2\sqrt{2}}\right)^{n} = \left(\frac{1}{\sqrt{2}}\right)^{n}$$

Thus, we have:

$$\Pr(X > n/2) < \left(\frac{1}{\sqrt{2}}\right)^{n}$$