Algorithm Design - Exam
16-06-2025, *Time: 120 minutes*
Prof. Stefano Leonardi, Dr. Federico Fusco

The first two problems correspond to the first partial exam, the third and the fourth to the second one.

Problem 1

A matching in an undirected graph is a set of edges with no common endpoints. The bipartite matching problem can be solved through a max-flow algorithm that repeatedly finds augmenting paths in a flow network.

(a) Formulate the problem of finding a bipartite matching of maximum cardinality as a max-flow problem and define the concept of augmenting path for the bipartite matching problem.

(b) Design an augmenting path based algorithm that solves optimally the bipartite matching problem and study its time complexity.

(c) Prove that the time complexity of the algorithm can be improved by selecting at each step the shortest augmenting path.

Problem 2

Let $G = (V, E)$ be a line graph of n nodes formed by a sequence of nodes $v_1, v_2, \ldots, v_n$ with an edge between v_i and v_j if and only if i and j differs by exactly 1. With every vertex i, we associate a positive integer weight w_i. Recall that a set of nodes is an *independent set* if no two nodes of them are joined by an edge. The goal is to find an independent set of G of maximum total weight.

(a) Show on an example that a greedy strategy fails to find an optimal solution.

(b) Give an algorithm that finds an independent set of maximum total weight in time which is polynomial in the number of vertices and it does not depend from the weights of the nodes.

(c) Provide the pseudo-code of an efficient implementation of the algorithm and discuss its time and space complexity.

Problem 3

(a) Give the formal definition of polynomial-time reduction from a decision problem X to a decision problem Y and prove the transitivity of such notion, i.e., if A is polynomial-time reducible to B and B is polynomial-time reducible to C, then A is polynomial-time reducible to C.

(b) Give the formal definition of efficient certifier for a decision problem X, and prove that $P \subseteq NP$.

(c) Define formally the independent set and vertex cover decision problems. Give an efficient certifier for the two problems and show a polynomial time reduction from one to the other, and vice-versa.

Problem 4

We are given a set of m machines $M_1, \ldots, M_m$ and a set of n jobs. Each job j has processing time t_j and is assigned to one of the machines. Denote by $A(i)$ the set of jobs assigned to machine i, and let $T_i = \sum_{j \in A(i)} t_j$ the load on machine M_i. The *load balancing* problem seeks to minimize the maximum load $T = \max_i T_i$ placed on a machine.

(a) Discuss whether the problem is NP-complete and define the concept of an approximation algorithm for the load balancing problem.

(b) Prove that the greedy algorithm that fixes an arbitrary order on the jobs and then assigns them to the machine which currently has the minimum load yields a 2 approximation. Prove the tightness of such approximation factor by providing an instance where the output of this greedy algorithm is *exactly* a 2 approximation of the optimal solution.

(c) Prove that the greedy algorithm that first orders the jobs by decreasing processing time, and then assigns the jobs in order to the machine which currently has the minimum load achieves a $3/2$ approximation.

Problem 1

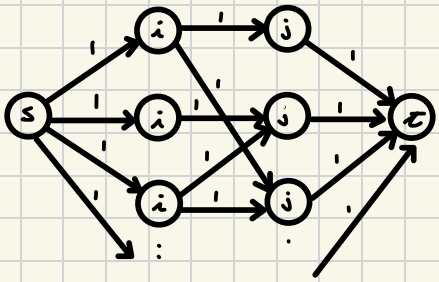
A matching in an undirected graph is a set of edges with no common endpoints. The bipartite matching problem can be solved through a max-flow algorithm that repeatedly finds augmenting paths in a flow network.

(a) Formulate the problem of finding a bipartite matching of maximum cardinality as a max-flow problem and define the concept of augmenting path for the bipartite matching problem.

(b) Design an augmenting path based algorithm that solves optimally the bipartite matching problem and study its time complexity.

(c) Prove that the time complexity of the algorithm can be improved by selecting at each step the shortest augmenting path.

a) WE CREATE G' BY ADDING A SOURCE s AND A SINK t LINKED WITH L AND R. WE CALCULATE THE MAX FLOW AND SO THERE IS A MAXIMUM MATCHING IFF THERE IS A MAX FLOW FROM s TO t. ALL EDGES HAVE CAPACITY 1.



AN AUGMENTING PATH IS A PATH THAT REORGANIZE MATCHING COUPLES IN THE GRAPH BY ALTERNING $e \in M$ AND $e \notin M$. IT STARTS AND ENDS AT UNMATCHED NODES. IT'S CALLED AUGMENTING BECAUSE IT ADDS A NEW MATCH.

b)

$M = \phi$
WHILE $\exists$ AN AUGMENTING PATH P: $\quad O(|M| \cdot |E|)$
  $M = M \oplus P$
RETURN M

THERE IS A MAXIMUM MATCHING IFF THERE AREN'T AUGMENTING PATHS.

c) THIS IS EDMONDS-KARP ALGORITHM.
EVERY AUGMENTING PATH FOUND IS A SHORTEST PATH FROM s TO t IN THE RESIDUAL GRAPH G'. AFTER EACH AUGMENTATION, THE DISTANCE FROM s TO ANY NODES IN G' NEVER DECREASES, AND THE DISTANCE FROM s TO t INCREASES AFTER AT MOST $O(m)$ AUGMENTATIONS, EACH OF THESE REQUIRES A BFS WITH $O(m)$ TIME.

$$O(m) \cdot O(n \cdot m) = O(nm^2)$$

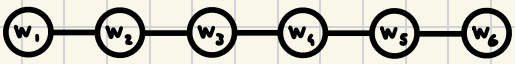
Problem 2

Let $G = (V, E)$ be a line graph of n nodes formed by a sequence of nodes $v_1, v_2, \ldots, v_n$ with an edge between v_i and v_j if and only if i and j differs by exactly 1. With every vertex i, we associate a positive integer weight w_i. Recall that a set of nodes is an *independent set* if no two nodes of them are joined by an edge. The goal is to find an independent set of G of maximum total weight.

(a) Show on an example that a greedy strategy fails to find an optimal solution.

(b) Give an algorithm that finds an independent set of maximum total weight in time which is polynomial in the number of vertices and it does not depend from the weights of the nodes.

(c) Provide the pseudo-code of an efficient implementation of the algorithm and discuss its time and space complexity.



a) A GREEDY ALGORITHM CAN CHOOSE ALWAYS THE NODE WITH THE MAXIMUM WEIGHT, AND THEN ELIMINATE ITS NEIGHBOURS (NOT GOOD):

IT WOULD CHOOSE w_2, BUT OPT = 8

b) OPT(i) = BEST SOLUTION USING $V_1 \ldots V_i$

$$V_i \begin{cases} OPT(i-1) & \text{DON'T CHOOSE } V_i \\ W_i + OPT(i-2) & \text{CHOOSE } V_i \end{cases}$$

$$OPT(i) = \max\left(W_i + OPT(i-2),\ OPT(i-1)\right)$$

c)

```
PATH-ALG

    IF n == 0   RETURN 0
    IF n == 1   RETURN w[1]
    OPT[0] = 0
    OPT[1] = w[1]

    FOR i = 2 TO n:
        OPT[i] = max(OPT[i-1], w[i] + OPT[i-2])
    RETURN OPT[n]
```

O(n)

Problem 3

(a) Give the formal definition of polynomial-time reduction from a decision problem X to a decision problem Y and prove the transitivity of such notion, i.e., if A is polynomial-time reducible to B and B is polynomial-time reducible to C, then A is polynomial-time reducible to C.

(b) Give the formal definition of efficient certifier for a decision problem X, and prove that $P \subseteq NP$.

(c) Define formally the independent set and vertex cover decision problems. Give an efficient certifier for the two problems and show a polynomial time reduction from one to the other, and vice-versa.

a) $X \leq_p Y$ IF $\exists$ A FUNCTION f CALCULABLE IN POLYNOMIAL TIME SUCH THAT FOR EACH INSTANCE $x \rightarrow x \in X \Leftrightarrow f(x) \in Y$.
WE TRANSFORM AN x INSTANCE IN A Y INSTANCE. SO IF WE CAN RESOLVE Y, WE CAN ALSO RESOLVE X.

$A \leq_p B$, $B \leq_p C \rightarrow$ $a \in A \Leftrightarrow f(a) \in B$, $b \in B \Leftrightarrow f(b) \in C$

LET $h(x) = g(f(x))$ $\quad a \in A \Leftrightarrow f(a) \in B \Leftrightarrow g(f(a)) \in C$

SO $a \in A \Leftrightarrow h(a) \in C$ AND $A \leq_p C$

b) C IS AN EFFICIENT CERTIFIER FOR X IF:

- C IS A POLYNOMIAL ALGORITHM THAT TAKES TWO INPUT s AND t.
- THERE IS A POLINOMIAL FUNCTION p S t.

$\forall$ STRING s, $s \in X$ IFF $\exists$ STRING t : $|t| \leq p(|s|) \wedge c(s,t) =$ YES

NP IS THE SET OF PROBLEMS FOR WHICH THERE EXISTS AN EFFICIENT CERTIFIER.
$X \in P$ MEANS THAT EXISTS AN ALGORITHM THAT SOLVES X IN POLYNOMIAL TIME. WE DEFINE A CERTIFIER $C(s,z)$ THAT IGNORES THE CERTIFICATE z AND SIMPLY RUNS $A(s)$. SINCE A RUN IN POL-TIME, C ALSO RUNS IN POL-TIME.

- IF $s \in X$, $A(s) =$ YES $\rightarrow$ $\exists$ CERTIFICATE z S.T. $C(s,z) =$ YES
- IF $s \notin X$, $A(s) =$ NO $\rightarrow$ $\forall$ CERTIFICATE z. $C(s,z) =$ NO

SO $X \in NP$ TOO, THEREFORE $P \subseteq NP$

c) IND SET : GIVEN A GRAPH $G=(V,E)$ AND AN INTEGER K, $\exists\, S \subseteq V$ S.T.
- $|S| \geq K$
- NO TWO NODES IN S ARE JOINED BY AN EDGE

S IS THE CERTIFICATE → VERIFY IF $|S| \geq K$ AND THAT THERE ARE NOT EDGES BETWEEN NODES IN S.

VER COV: GIVEN A GRAPH $G=(V,E)$ AND AN INTEGER K, $\exists\, C \subseteq V$ S.T.
- $|C| \leq K$
- $\forall e \in E.$ e HAS AN END IN C

C IS THE CERTIFICATE → VERIFY IF $|C| \leq K$ AND THAT EVERY EDGE HAS AN END IN C.

S IS AN INDIPENDENT SET $\Leftrightarrow$ $V-S$ IS A VERTEX COVER

$\Rightarrow$ IND - VER

INSTANCE : (G, K)

WE CONSTRUCT $(G, |V|-K)$

THERE IS AN INDIPENDENT SET OF SIZE AT LEAST K IFF THERE IS A VERTEX COVER OF SIZE AT MOST $|V|-K$.

$\Leftarrow$ VER - IND

INSTANCE : (G, K)

WE CONSTRUCT $(G, |V|-K)$

THERE IS A VERTEX COVER OF SIZE AT MOST $|V|-K$ IFF THERE IS AN INDIPENDENT SET OF SIZE AT LEAST K.

Problem 4

We are given a set of m machines $M_1, \ldots, M_m$ and a set of n jobs. Each job j has processing time t_j and is assigned to one of the machines. Denote by $A(i)$ the set of jobs assigned to machine i, and let $T_i = \sum_{j \in A(i)} t_j$ the load on machine M_i. The *load balancing* problem seeks to minimize the maximum load $T = \max_i T_i$ placed on a machine.

(a) Discuss whether the problem is NP-complete and define the concept of an approximation algorithm for the load balancing problem.

(b) Prove that the greedy algorithm that fixes an arbitrary order on the jobs and then assigns them to the machine which currently has the minimum load yields a 2 approximation. Prove the tightness of such approximation factor by providing an instance where the output of this greedy algorithm is *exactly* a 2 approximation of the optimal solution.

(c) Prove that the greedy algorithm that first orders the jobs by decreasing processing time, and then assigns the jobs in order to the machine which currently has the minimum load achieves a $3/2$ approximation.

a) THE DECISIONAL VERSION IS NP-COMPLETE: B = GIVEN BOUND

THERE EXISTS A JOB ASSIGNMENT TO m MACHINES S.T. $T \leq B$?

AN ALGORITHM IS A ρ-APPROX IF FOR EACH INSTANCE IT PRODUCES A SOLUTION WITH COST $T \leq \rho \cdot OPT$, WHERE OPT IS THE MINIMUM POSSIBLE MAKESPAN.

b) OPT = OPTIMAL MAKESPAN

WE HAVE TWO LOWER BOUND

1. $OPT \geq \frac{\sum_j t_j}{m}$ TOTAL WORK HAS TO BE DISTRIBUTED ON m MACHINES
2. $OPT \geq \max_j t_j$ A JOB HAS TO BE ENTIRELY ON A MACHINE

LET j BE THE LAST JOB ASSIGNED ON THE LAST m. T_{ALG} IS THE ALG'S MAKESPAN, t_j, j'S DURATION, AND L THE m'S LOAD BEFORE j.

$T_{ALG} = L + t_j$ $\quad L \leq \frac{\sum_j t_j}{m}$ AVERAGE LOAD IN THIS MOMENT

$T_{ALG} \leq \frac{\sum_j t_j}{m} + t_j$ $\quad T_{ALG} \leq OPT + OPT = 2OPT$

---

m MACHINES $\quad m(m-1)$ JOB WITH $t_i = 1$ $\quad$ ONE JOB WITH $t = m$

IF JOB WITH $t_i = 1$ ARRIVE FIRST, THE GREEDY DISTRIBUTES THEM ON m MACHINES TILL A LOAD m-1. WHEN THE BIGGER JOB ARRIVES IT GOES ON A MACHINE (ALL EQUAL) AND THE MAXIMUM MAKESPAN IS:

$$T_{ALG} = (m-1) + m = 2m - 1$$

BUT $OPT = m$, BY PUTTING THE BIGGER JOB ON A MACHINE LONELY.

$T_{ALG}/OPT = \frac{2m-1}{m} = 2 - \frac{1}{m}$ TENDS TO 2 WHEN m GROWS.

c) T = ALG'S MAKESPAN T^* = OPT MAKESPAN

WE HAVE $\tau_1 \geq \tau_2 \geq \dots \geq \tau_m \geq \tau_{m+1}$

FOR SURE A MACHINE WILL HAVE AT LEAST 2 JOBS ASSIGNED

SO $T^* \geq 2\tau_{m+1} \rightarrow \tau_{m+1} \leq \frac{1}{2}T^*$

LET M_i BE THE MACHINE WITH MAXIMUM LOAD, THEN $T = T_i$.
THERE 2 CASES:

- M_i HAS ONLY ONE JOB, THE BIGGEST $\rightarrow T = \tau_1 \leq T^*$
- M_i HAS AT LEAST 2 JOBS:

  LET j BE THE LAST JOB ASSIGNED TO M_i

  $j \geq m+1 \rightarrow \tau_j \leq \tau_{m+1} \rightarrow \tau_j \leq \frac{1}{2}T^*$

  M_i'S LOAD IS $T_i = (T_i - \tau_j) + \tau_j$

  BEFORE τ_j, M_i WAS THE m WITH MINIMUM LOAD

  $T_i - \tau_j \leq T^*$ T^* CAN'T BE $\leq$ MINIMUM LOAD DURING THE ASSIGN.

  $T_i = (T_i - \tau_j) + \tau_j \rightarrow T_i \leq T^* + \frac{1}{2}T^* \rightarrow T_i \leq \frac{3}{2}T^*$