COL351 Fall 2021 Major Exam

Tamajit Banerjee

TOTAL POINTS

29.5 / 30

QUESTION 1

Short Questions 8 pts

1.1 (a) 3 / 3

✓ **+ 3 pts Correct**

+ 1 pts Proof of NP class is correct.

+ 2 pts Only giving a reduction, but not proving that problem is in NP class.

+ 1.5 pts Giving a reduction, but not providing all reduction properties of NP-hard.

+ 1 pts Incomplete reduction.

+ 0.5 pts Stating some facts about NP-completeness, or
right direction to NP-completeness proof,
but the attempted reduction is incorrect/incomplete (though in the right direction).

Note - No marks for completely wrong reduction.

+ 0 pts -- Incorrect, or
-- Not attempted, or
-- Only stating facts about NP-completeness / Independent set / 3SAT Problem but no reduction presented.

1.2 (b) 3 / 3

✓ **+ 3 pts Both parts are correct**

+ 1 pts Showing that if G is acyclic, then T is a DFS tree

+ 2 pts Showing that if T is DFS tree, then G is acyclic.

+ 1.5 pts The part that T is DFS tree, implies G is acyclic lacks clarity or is partially incorrect.

Need to provide correct argument either using some properties of DFS traversal, or the facts that
-- Non-tree edges wrt DFS tree can only be back edges (i.e. cannot be cross edges), and
-- The difference in the level of endpoints of an edge in a BFS tree is at most one.

+ 0 pts Incorrect / Not attempted.

+ 1 pts -- Non-tree edges wrt DFS tree can only be back edges (i.e. cannot be cross edges), and

1.3 (c) 2 / 2

✓ **+ 2 pts Correct**

+ 1 pts partial solution. Only product of two polynomials computed.

+ 0 pts Incorrect / Not attempted

QUESTION 2

Divide and Conquer 9 pts

2.1 (a) 5.5 / 6

✓ **+ 1 pts Array defined for size 100n to account for all sums.**

✓ **+ 3 pts Divide and Conquer Steps**

+ 0 pts Incorrect / Unattempted

✓ **+ 1 pts Correctness**

✓ **+ 1 pts Time Complexity Details**

- 0.5 Point adjustment

Correctness partially correct

2.2 (b) 3 / 3

✓ **+ 3 pts Correct**

+ 0 pts Incorrect / Unattempted

QUESTION 3

3 Max-Flow Application 6 / 6

+ 0 pts Incorrect / did not attempt

✓ **+ 1 pts Correct structure of the graph**

✓ **+ 0.5 pts Correct edge capacities: supply side**

✓ + 0.5 pts Correct edge capacities: demand side

✓ + 1 pts Correct edge capacities: donor/receiver constraints

✓ + 1 pts Correct theorem statement: demand can be met iff max flow is $$d_O + d_A + d_B + d_{AB}$$ (or equivalently flow through each demand-side node matches the corresponding demand).

✓ + 2 pts Proof: integrality of flow not argued.

+ 0 pts Proof: argued integrality of max-flow.

QUESTION 4

4 Profit Optimization 7 / 7

✓ + 4 pts Correct DP equation

✓ + 1 pts Correct boundary cases

✓ + 2 pts Correctness argument

+ 0 pts Incorrect / Not attempted

+ 2 pts Partially correct answer

1 Incorrect

QUESTION 5

5 Scan of Question Paper 0 / 0

✓ + 0 pts Correct

Tamajit Banerjee 2019CS10408

1.) Short Questions

(a) To show a problem is NP complete, we can show two properties

- The problem is in NP class
- A problem which is NP complete reduces to the problem.

1. or ~~We can prove that the problem~~ If we are given a solution for the problem, we can verify the following

- $|S| \leq k$ This will take $O(n)$ time as The solution
- We can take all pair of vertices and check if each of the pairs have an edge between them or not. Mathematically

  $\forall (i,j) \in S \Rightarrow$ there should be an edge e between i and j.

  $O(n^2)$

$\Rightarrow$ We can verify if a given certificate is a solution or not in polynomial time

2. We can reduce Independent Set to the given problem (Let it be called X)

∴ Independent set $\leq_p$ X

∵ Independentset is NP-complete (as proved in lecture)

⇒ X is also NP-complete

Transforming an instance of Independent set to X.

Given: a Graph G(V, E) and k, we have to find if there is an Independent Set of size k.

we make a new graph H(V, $\bar{E}$) where the edges are complement of the original graph.

⇒ if $(i,j) \notin E \Leftrightarrow (i,j) \in \bar{E}$.

Here, we need to find a nice set of size k.

The time to convert G to H is O( ) which is polynomial in input size

Proof: an "yes" instance of Independ Set ⇒ "yes" instance of X.

Let S be the solution of Independent set of size k

⇒ S is also a solution of X

∴ $\forall (i,j) \in S$, there is no edge in G = (i,j)

⇒ in the graph H, we will have $\forall (i,j) \Leftrightarrow$ there is an edge in H = (i,j)

Proof : an "yes" instance of X ⇒ an yes instance of independent set

a solution S of X ⇒ it is also a solution of independent set

i.e. $\forall (i,j) \in S$, there is an edge (i,j) in H

⇒ $\nexists (i,j) \in S$, there is no edge in G as complement

Hence proved

⇒ we have converted I S to X in polynomial time

+ proved that "yes" instance

1.1 (a) 3 / 3

✓ **+ 3 pts Correct**

+ 1 pts Proof of NP class is correct.

+ 2 pts Only giving a reduction, but not proving that problem is in NP class.

+ 1.5 pts Giving a reduction, but not providing all reduction properties of NP-hard.

+ 1 pts Incomplete reduction.

+ 0.5 pts Stating some facts about NP-completeness, or
right direction to NP-completeness proof,
but the attempted reduction is incorrect/incomplete (though in the right direction).

Note - No marks for completely wrong reduction.

+ 0 pts -- Incorrect, or
-- Not attempted, or
-- Only stating facts about NP-completeness / Independent set / 3SAT Problem but no reduction presented.

1.) (b)

Proof:

G is acyclic ⇒ T is also DFS tree of G

Proof by implication

∴ G is acyclic and G is connect and undirected

⇒ T is a tree and it has n-1 edges.

[theorem done in class] [prove by induction]

∴ T is a tree, ⇒ there is only a unique tree

⇒ BFS tree = DFS tree

Proof:

T is a DFS tree ⇒ G is acyclic

Proof by contradiction

suppose G is cycle, ⇒ there exists a cycle in G let it be C, $v_1, v_2, \ldots v_k$.

Suppose

v_1 be the first vertex reached by BFS algo

then the tree calculated by it will have v_2 and v_k as its neighbours

Now, v_1 be the first vertex reached by DFS algo

when we run dfs, we will either visit v_2 or v_k $\Rightarrow$ and if the either will visit the other one so that would not be a neighbour of 1

$\therefore$ reachable $\Rightarrow$ descendant $\therefore$ reachable pts will

$\Rightarrow$ DFS tree $\neq$ BFS tree

$\Rightarrow$ It is a contradiction to our assumption.

Hence proved

1.2 (b) **3 / 3**

✓ **+ 3 pts** **Both parts are correct**

+ 1 pts Showing that if G is acyclic, then T is a DFS tree

+ 2 pts Showing that if T is DFS tree, then G is acyclic.

+ 1.5 pts The part that T is DFS tree, implies G is acyclic lacks clarity or is partially incorrect.

Need to provide correct argument either using some properties of DFS traversal, or the facts that

-- Non-tree edges wrt DFS tree can only be back edges (i.e. cannot be cross edges), and

-- The difference in the level of endpoints of an edge in a BFS tree is at most one.

+ 0 pts Incorrect / Not attempted.

+ 1 pts -- Non-tree edges wrt DFS tree can only be back edges (i.e. cannot be cross edges), and

(c)

$A = [a_1, a_2, \cdots, a_n]$

$B = [b_1, b_2, \cdots, b_n]$

Let a define a polynomial

$$AP(x) = A_0 x^0 + A_1 x^1 + \cdots + A_i x^i + \cdots + A_{100n} x^{100n}$$

where the coefficient A_i of x^i = 1 if there exists $a \in A$ such that $a = i$

exists = 0 otherwise

Similarly,

$$BP(x) = B_0 x^0 + B_1 x^1 + \cdots + B_i x^i + B_{100n} x^{100n}$$

where the coefficient B_i of x^i = 1 if there exists $b \in B$ such that $b = i$

0 otherwise.

We define,

$$ZP(x) = AP(x) * B(x)$$

$$\Rightarrow ZP(x) = Z_0 x^0 + Z_1 x^1 + \dots + Z_i x^i + \dots + Z_{200n} x^{200n}$$

where Z_i where $Z_i = \left(\sum_{j=0}^{i} A_j B_{i-j} \right)$

$\Rightarrow$ if the coefficient of $x^i = Z_0^i > 0 \Rightarrow$

there exists $a \in A$ and $b \in B$

such that $a+b = i$

and $i <= 200n$ as $a <= 100n$ and $b <= 100n$

We know from the polynomial multiplication of two polynomials $A(x)$ and $B(x)$ whose highest degreen is n, takes $O(n \log n)$ time by using Fast Fourier transform (DFT, IDFT)

$\Rightarrow$ To calculate $ZP(x)$ we take $O(200n \log(200n))$

$\Rightarrow ZP(x) = O(n \log n)$ removing constant

$Z_i > 0 \Rightarrow i \in Z$

1.3 (c) **2 / 2**

✓ **+ 2 pts Correct**

+ 1 pts partial solution. Only product of two polynomials computed.

+ 0 pts Incorrect / Not attempted

2(a) Algorithm

Global variable & set $S = \{\}$

Calculate-set (C, i, j)

if (i == j):
  S.insert (C[i]) # equivalent to $S \leftarrow S \cup \{C[i]\}$
  return

$m \leftarrow \frac{(i+j)}{2}$

Calculate-set (C, i, m)

Calculate-set (C, m+1, j)

~~Calculate~~ create two arrays A [0 ... m-i+1]
and B [0, ... j-m]
Initialise all the elements to 0.

~~Creates two~~

sum_A ← 0

for l ← 0 to m-i+1
  sum_A += C [m-l]
  A[l] ← sum_A

sum_B ← 0

for l ← 0 to j-m
  sum_B += C [m+1+j]
  B[l] ← sum_B

Now, use the algorithm given in 1(c)

to use (A and B) as input and

return the set Z.

$S \leftarrow S \cup Z$

return

Correctness

Assuming Calculate-set (C, i, m) correctly

calculates all sum in the array C[i ... m]

and Calculate set (C, m+1, j) correctly

calculate all sum in the array C[m+1 ... j]

A[l] stores the sum of C[m, ... m-l]

B[l] store the sum C[m+1, ... m+1+l]

⇒ A * B computes all the sum

as the sum between these two arrays

will be some part of the left and

some part of the right

⇒ Z is computed correctly

Base case is true

The 1st step to complete

A takes $O(n/2)$ time

and B takes $O(n/2)$ time

as element of A and B $<-(n/2 \times 100)$

as each elem of C $<= 100$

$\Rightarrow$ A * B takes $O(n \log n)$ time

and the calculate_set $(C, i, m) \leftarrow T(n/2)$

calculate_set $(C, m+1, j) \leftarrow T(n/2)$

$\Rightarrow$ relation

$$T(n) = T(n/2) + T(n/2) + O(n/2) * 2 + O(n \log n)$$

$$= 2T(n/2) + O(n \log n)$$

2.1 (a) **5.5 / 6**

✓ **+ 1 pts** Array defined for size 100n to account for all sums.

✓ **+ 3 pts** Divide and Conquer Steps

+ 0 pts Incorrect / Unattempted

✓ **+ 1 pts** Correctness

✓ **+ 1 pts** Time Complexity Details

- 0.5 Point adjustment

Correctness partially correct

(5) Let 2^x be the first power of 2 greater than or equal to n

$\Rightarrow 2^x \geq n$ but $2^x < 2 \cdot n$.

if not then $x-1$ would have been the 1st power

Putting 2^x in the equation, we get

$T(2^x) = 2T(2^{x-1}) + O(2^x \cdot x)$

We can assume $O(2^x \cdot x) \leq c \cdot 2^x \cdot x$ for some $c \geq 1$

$\Rightarrow T(2^x) = 2T(2^{x-1}) + c \cdot 2^x \cdot x$

$= 2^2 T(2^{x-2}) + c \cdot 2^x \cdot (x-1) + c \cdot 2^x \cdot x$

$= 2^3 T(2^{x-3}) + c \cdot 2^x (x + (x-1) + (x-2))$

Expanding like this, we get

$= 2^x T(1) + c \cdot 2^x (x + (x-1) + (x-2) + \dots + 1)$

$= 2^x T(1) + c \cdot 2^x \cdot \frac{x(x+1)}{2}$

$$\Rightarrow T(2^x) = 2^x T(1) + c_0 2^x \cdot \frac{x(x+1)}{2}$$

we know, $T(1) = O(1)$

$$\Rightarrow T(2^x) = 2^x O(1) + 2^x \frac{x(x+1)}{2} \; O(1)$$

~~$2^x \left(x(x+1) \right)$~~

Substituting $x = \lceil \log_2 n \rceil$

and $x \leq 2 \cdot n$

$$\Rightarrow T(2^x) \leq 2 \cdot n \, O(1) + 2 \cdot n \, \frac{\log_2 n (\log_2 n + 1)}{2} \; O(1)$$

$$= O(2n) + O(n \log_2^2 n)$$

$$\therefore T(n) \leq T(2^x) = O(n \log^2 n)$$

$$\Rightarrow T(n) = \underline{O(n \log^2 n)}$$

Hence proved

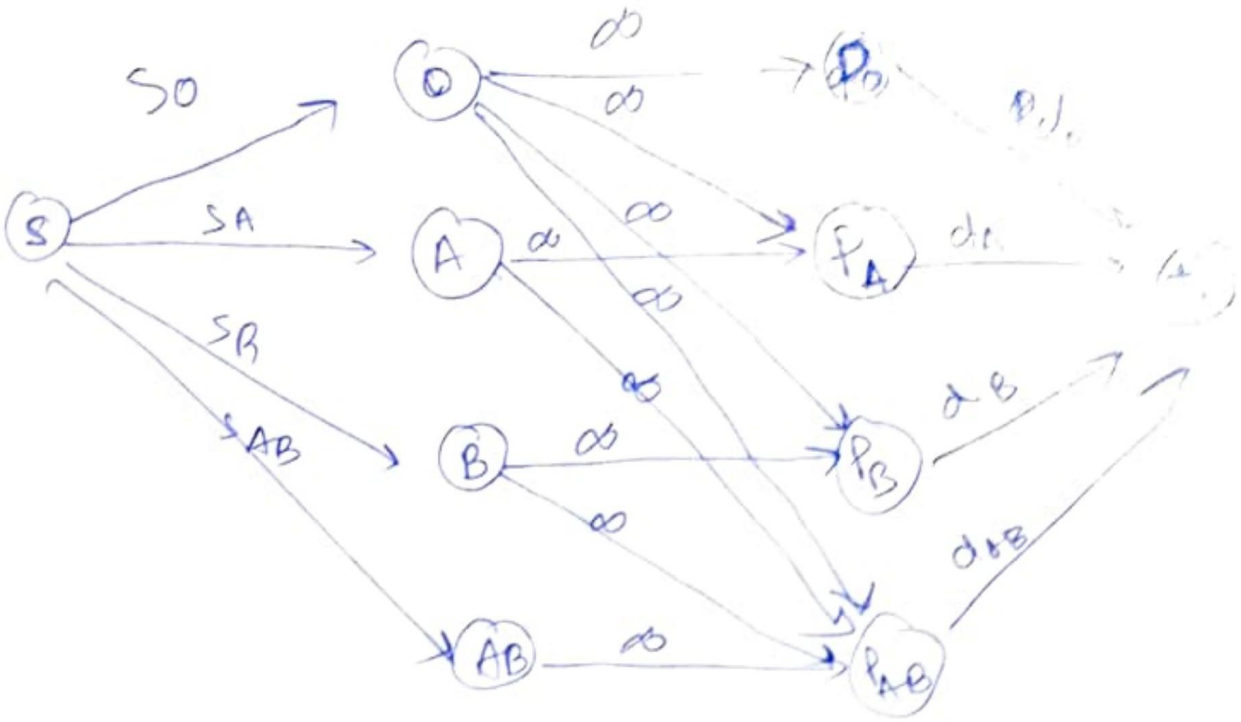
2.2 (b) **3 / 3**

✓ **+ 3 pts** **Correct**

+ 0 pts Incorrect / Unattempted

3.)

Let us define, the following graph,



- We ~~used~~ use S as source and t as sink.
- Added edges from S to A, B, O, AB with capacities S_A, S_B, S_O and S_{AB}
- Added edge from P_A, P_B, P_{AB}, P_O to t with the demands d_A, d_B, d_{AB}, d_O
- Added edges from O to P_O, P_A, P_B, P_{AB} as O can supply to those with edge capacities = ∞ and E for others.

Algorithm

Supply(

① Define a graph (G) as shown in the diagram.

② Compute the s-t max flow in the graph using Ford-fulkerson Algorithm (done in class)

③ If max f < $d_0 + d_A + d_B + d_{AB}$

return false

else

return true [i.e. f = $d_0 + d_A + d_B + d_{AB}$])

Proof of correctness

① If there exists a way to distribut the blood to meet the demand, we can create a s-t max flow of flow f

Proof: by construction

② Suppose we send amount of O will go to P_O y to P_B, z to P_A, l to P_{AB}, we can supply that amount of flow to them as $x+y+z+l \leq S_O$

Similarly, we can do it for other blood groups as well.

and in the end each of the P_i will receive d_i amount of blood as given in the question.

$\Rightarrow$ we can divide this flow to t

and we get the max-flow = Σd_i

$\Rightarrow$ max-flow of $\Sigma d_i \Rightarrow$ supply-demand met

Proof: $\Rightarrow$ the d_i edges are in min cut, i.e they are fully saturated $\Rightarrow$ at least d_i amount of flow reached to $P_i \Rightarrow$ the bloods groups were able to supply that amount of flow to each.

Hence proved

3 Max-Flow Application 6 / 6

+ **0 pts** Incorrect / did not attempt

✓ **+ 1 pts** **Correct structure of the graph**

✓ **+ 0.5 pts** **Correct edge capacities: supply side**

✓ **+ 0.5 pts** **Correct edge capacities: demand side**

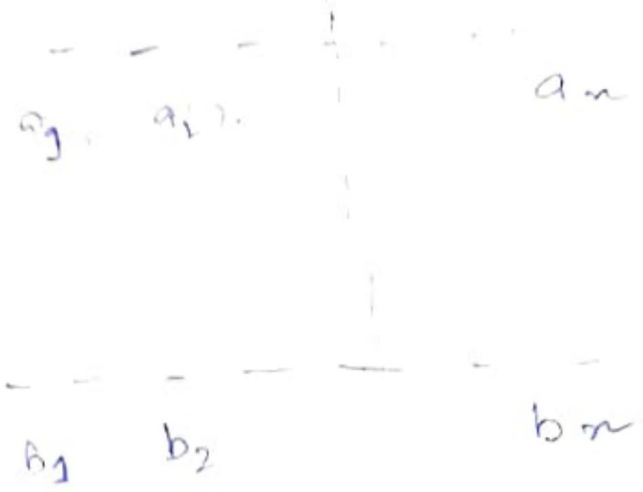
✓ **+ 1 pts** **Correct edge capacities: donor/receiver constraints**

✓ **+ 1 pts** **Correct theorem statement: demand can be met iff max flow is $$d_O + d_A + d_B + d_{AB}$$ (or equivalently flow through each demand-side node matches the corresponding demand).**

✓ **+ 2 pts** **Proof: integrality of flow not argued.**

+ **0 pts** Proof: argued integrality of max-flow.

4.)



dp[i] ← maximum profit gained by connecting i and M(i) if it is possible to connect

~~i and j ie j = M(i)~~

dp[i] = 0 ∀ i, j

we insert M(0) = 0.

Q.1

dp[i] = maximum profit gained by connecting i and M(i) if it is possible to connect

~~i and j = M(i)~~

dp[i] = 0 ∀ i, j

We insert M(0) ...

Algorithm

dp[0, n+1] = 0

M ← M ∪ {0 → 0}.. M(0) = 0

for i ← 1 to n

for j ← 0 to i-1

~~dp[i] = max (dp[i], dp[~~

if M(j) ≤ M(i):

dp[i] = max (dp[i], dp[j] + P[i])

return the maximum of dp[0,.. n]

Proof time complexity

① it takes $O(n^2)$ time to do for-loop

② it takes $O(n)$ to initialize and return the maximum ⇒ total time $O(n)$

Proof of correctness

Base case:

dp[0] = 0

and M(0) = 0 as there is no inversion due to the

recursion:

$$dp[i] = \min[dp[j] + P[i]] \text{ where } M[j] < M[i]$$

⇒ ⇒ ⇒ no inversion

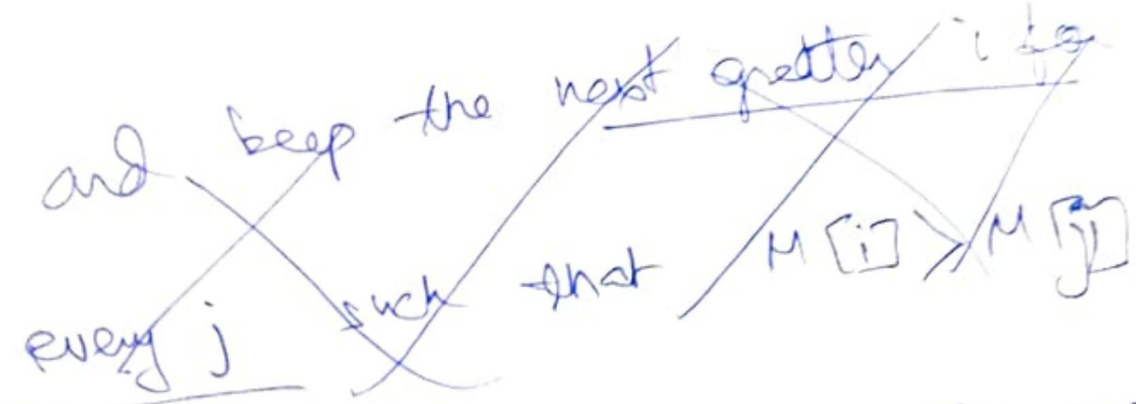
Hence

since we calculate in the increasing order of i i.e. 0, ..., i, we have the correct value of dp[j] ∀ j < i and have dp[i] is also correct by strong induction.

Can be calculated in $O(n \log n)$

by changing the dp. slightly

and keep the next greater i for every j such that $M[i] > M[j]$



we are able to reach for max $dp[i]$ in $(\log n)$

this can be done by storing sorted $dp[j]$ for $\forall j < i$ and the $M[j]$ are in increasing order

4 Profit Optimization 7 / 7

- ✓ **+ 4 pts** **Correct DP equation**
- ✓ **+ 1 pts** **Correct boundary cases**
- ✓ **+ 2 pts** **Correctness argument**
- **+ 0 pts** Incorrect / Not attempted
- **+ 2 pts** Partially correct answer

1 Incorrect

Online COL 351 Major Exam

Timing: 1:00 PM - 2:30 PM

Important Guidelines:

1. This is a closed book exam. You cannot look at your notes or browse the internet.
2. You can directly reference an algorithm / theorem proved in the lectures. However, results proved in tutorials cannot be directly used in Major exam.
3. Your answer to each question must be formal and have a correctness proof.
4. Total marks are 30.
5. The scanned solution **must be** uploaded on Gradescope by 2:45 PM sharp. The late submission deadline is 2:50 PM, and NO submissions will be accepted later under any circumstance.

1. Short Questions [3 + 3 + 2 = 8 marks]

(a) A subset S of vertices in a graph is said to be *nice* if every two distinct vertices in it are adjacent. Consider the following problem: Given an undirected graph G and an integer k, does G contain a nice set of size k? Show that this problem is NP-complete. Clearly state all the properties/facts you use to prove your claim.

(b) Let $G = (V, E)$ be an undirected connected graph, $s \in V$ be a vertex, and T be a BFS tree of G rooted at s. Prove that G is acyclic iff T is also a DFS tree of G.

(c) Let $A = [a_1, \ldots, a_n]$ and $B = [b_1, \ldots, b_n]$ be two arrays of positive integers in range $[1, 100n]$. Prove that the set $Z = \{a + b \mid a \in A,\ b \in B\}$ is computable in $O(n \log n)$ time.

2. Divide and Conquer [6 + 3 = 9 marks]

(a) Let $C = [c_1, c_2, \ldots, c_n]$ be an array with all entries in range $[1, 100]$. Use 1(c) to obtain a divide and conquer algorithm to compute the set

$$S = \{C[i] + C[i+1] + \cdots + C[j] \mid i \leq j \in [1, n]\}$$

Argue that the time-complexity of your algorithm follows the relation $T(n) = 2T(n/2) + O(n \log n)$.

(b) **Without** using Master's theorem prove that the recurrence $T(n) = 2T(n/2) + O(n \log n)$ satisfies the relation $T(n) = O(n \log^2 n)$. You can assume that $T(0), T(1)$ are $O(1)$.

3. Max flow Application [6 marks] The basic rule for blood donation are: A patient of blood group A can receive only blood of group A or O. A patient of blood group B can receive only blood of group B or O. A patient of blood group O can receive only blood of group O. A patient of blood group AB can receive blood of any group.

Let s_O, s_A, s_B, s_{AB} denote the supply in whole units of the different blood types in a hospital for the coming week. Assume that the hospital knows the projected demand for each blood type d_O, d_A, d_B, and d_{AB} for the coming week. Give a max-flow based algorithm to check if the supply would meet the projected demand.

4. Profit Optimization [7 marks] Consider a 2-D map with a river passing parallel to x-axis in east-west direction. There are n factories on the northern bank, namely, $a_1, \ldots, a_n$ and n suppliers on the southern bank, namely, $b_1, \ldots, b_n$. Further, it is given that the x-coordinate of a_i and b_i is i. You are given a bijective map $M : [1, n] \to [1, n]$ such that factory a_i requires to be connected to supplier $b_{M(i)}$.

Let $P : [1, n] \to \mathbb{R}^+$ be a function such that $P(i)$ denotes the profit of connecting a_i with $b_{M(i)}$. Your task is to connect factory and corresponding suppliers with non-crossing bridges so as to maximize the sum total profit. Design a polynomial time algorithm to achieve the same.

5. Question paper Scan Take a picture of exam paper on your computer and upload it.

5 Scan of Question Paper 0 / 0

✓ **+ 0 pts** **Correct**