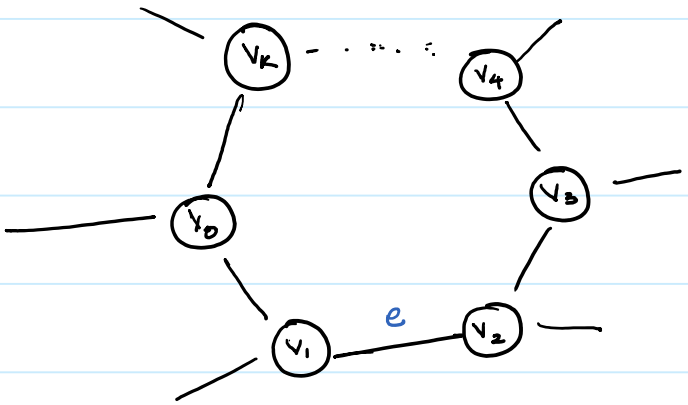
Problem 1 (25 points): Let e be a maximum-weight edge on some cycle of a connected, weighted, undirected graph G(V,E). Prove that there exists a minimum spanning tree of G that does not include e.

IDEA: As it is the maximum weight edge, for any cut across cycle, there exists atleast one edge lighter than this and it can be included in MST

By theorem: For an undirected graph $G(V,E)$. Let A be a subset of E that is included in some minimum spanning tree for G, let $(S, V-S)$ be any cut of G that respects A and let (u,v) be a light edge crossing $(S, V-S)$. Then edge (u,v) is safe for A.

Now for a cycle in connected graph.

Let $c = \langle v_0, v_1, v_2, \ldots, v_k \rangle$ form cycle in G



Without loss of generality assume (v_1, v_2) is max weight edge e

Now define a cut $(V', V-V')$ in the graph $G(V,E)$ such that

$[v_1 \in V' \text{ and } v_2 \in V-V']$

For any A this cut respects. [No edge in A crosses the cut]

By the theorem,

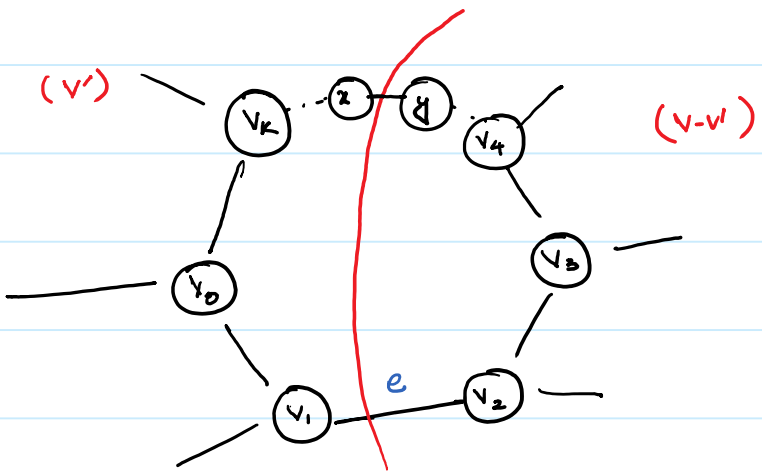
As $(V', V-V')$ respects A, then a light edge crossing this cut is a safe edge.

As the graph is connected and edge (v_1, v_2) is part of cycle

$\exists$ some edge (x, y) where $x, y \in V$ s.t. $(x, y) \neq (v_1, v_2)$ that crosses the cut defined by $(V', V-V')$

As edge (v_1, v_2) $\boxed{e}$ is the maximum weight edge

$w(v_1, v_2) > w(x, y)$ $\forall$ $x, y \in V$ s.t. $(x, y) \neq (v_1, v_2)$



By theorem, we choose edge light edge (x, y) than (v_1, v_2)

Thus $\exists$ MST which does not include the maximum weight edge $\boxed{e}$

Problem 2 (25 points): Assume that you own an office that you would like to rent for short time meetings, and you wish to maximize your rental income. Your plan is to charge a fixed fee per meeting, regardless of the duration of a meeting. Your customers provide you with a start time for their meeting and the duration of the meeting, in minutes. Thus, the input consists of two integers per customer, one for the start time and one for the duration of the meeting. You also assume a 10 minutes gap time between two consecutive meetings: the meeting that ends needs to vacate and clean the office before the next meeting can start. Assume you have a total of n customers each day and the scheduling is done for one 24 hours day (e.g., you select customers for the next day). Prove that the greedy choice of selecting the meeting with the latest start time leads to an optimal solution.

$s(i)$ → START TIME , $d(i)$ → DURATION , [10 min gap b/w two consecutive meetings]

Total of n customers, SCHEDULING for one 24 hrs day.

So, the meetings can be described as follows.

$c(i) : \langle s(i), s(i) + d(i) + 10 \rangle$

[START TIME] [FINISH TIME]

GREEDY CHOICE : Select the meeting with the latest start time

This greedy choice of selecting the start time leads to an optimal solution

It can be proved as follows.

For a non empty subproblem S_k, let a_m be a meeting in S_k with latest start time. Then a_m is included in some maximum size subset of mutually compatible meetings in S_k

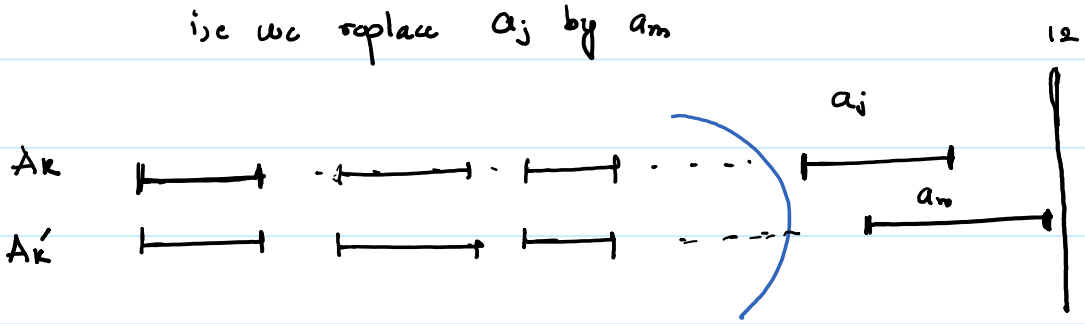
$\Rightarrow$ Let A_k be a maximum size subset of mutually compatible meetings in S_k and let a_j be the meeting in A_k with latest start time.

If $a_m = a_j$ [DONE] (Choice is included)

else

we define the set $A_k' = A_k - \{a_j\} \cup \{a_m\}$

i.e we replace a_j by a_m



Meetings in A_k' are compatible and disjoint follows because

A_k meetings are disjoint & $[s(j) < s(m)]$, $|A_k'| = |A_k|$

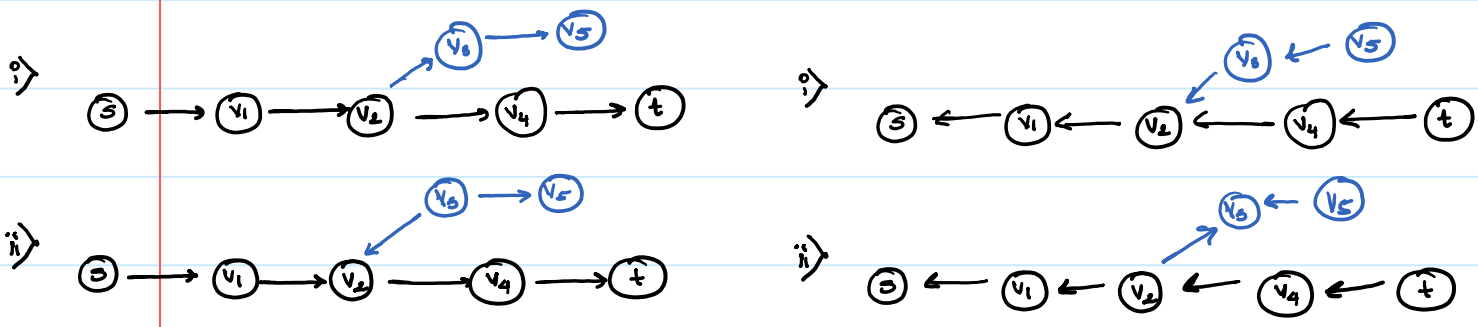
$\Rightarrow$ A_k' is maximum size subset of mutually compatible meetings of S_k and includes a_m

Problem 3 (25 points): In the maximum flow problem (single source single sink version), we assume that each vertex v of the flow network G(V,E) is on some path from the source s to the sink t. Give a linear time algorithm (in the size of the graph) to identify and remove vertices that do not satisfy this condition. Clearly describe your algorithm and argue its correctness.
Assume that after this process you are left with a flow network G' with n vertices and 7n-20 edges. What would be the running time of Edmond-Karp's maximum flow algorithm for G', as a function of n?

INTUITION

A vertex (v) if it is not present in any path from (s) to (t)

Then, it can be of following possibilities

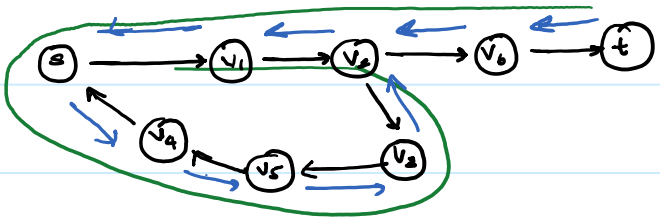


ALGORITHM

O(V+E) (1) List all the vertices reachable from source (s)

O(V+E) (2) Reverse edges of the graph, List all the vertices reachable from (t)

Vertices in the intersection of both lists are the ones that lie in path from s to t



* interesting case

PROOF →

CORRECTNESS OF ALGORITHM

Proof by exhaustion → [4 Cases]

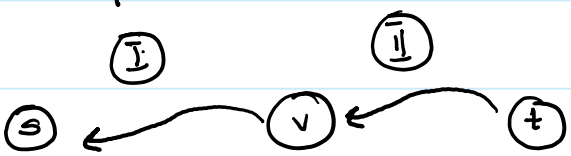
PART-I

let v be the vertex that is present in the path from (s) to (t)

but is not identified by the algorithm

By first step of algorithm (v) should be reached from (s)

By second step



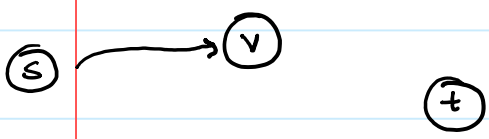
(v) should be reached from (t)

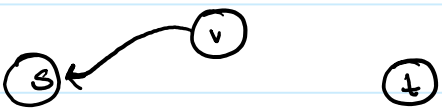
So, algorithm has [v] in its combined list [CONTRADICTION]

PART-II

let v be the vertex that is reachable from s

and it is not present on path from s to t





First list of algorithm has [v]

Second list does not have v as it is not reachable from (t)

PART-III

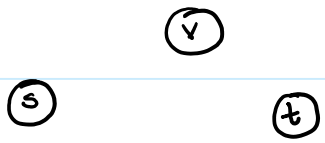
let v be the vertex not reachable from s

Step I of algorithm does not have this vertex in its list

G

PART-IV

let v be vertex not reachable from s, t in G



Step -I of algorithm does not have this vertex in its list

Thus in (4) cases

1. s → v → t
2. s → v t
3. s v → t (or) s v ← t
4. s v t

Algorithm correctly identifies the vertices

B) $n \rightarrow$ Vertices, $7n-20 \rightarrow$ edges.

Running Time of EDMOND-KARP's algorithm $O(VE^2)$

$$= O\left(n(7n-20)^2\right) = O(n^3)$$

(3)

G(V, E)

boolean reached1[ ] = DFS (G, s) (O(V+E))

Reverse (G)

boolean reached2[ ] = DSF (G, t) O(V+E)

for i = 1 to n

if reached1 [i] == reached2[i] O(V)

then add i to list

Problem 4 (25 points): The Subset Sum problem is defined as follows: Given numbers $x_1, x_2, \ldots, x_n$ and t, all > 0, is there a subset of $x_1, x_2, \ldots, x_n$ which sum to t?
The Partition problem is defined as follows: Given a set of numbers $X=\{x_1, x_2, \ldots, x_k\}$, can X be partitioned into two subsets A and B such that $A \cup B = X$, $A \cap B = \emptyset$, and $\sum_{xi \in A} xi = \sum_{xi \in B} xi$?
Assume that $t < (x_1 + x_2 + \ldots + x_n)/2$.
Prove that Partition is NP-hard by reduction from Subset Sum, with the assumption that the input for Subset Sum has $t < (x_1 + x_2 + \ldots + x_n)/2$.

Given a set of numbers $X = \{x_1, x_2, x_3 \ldots x_n\}$ can x be partitioned into A and B such that $A \cup B = X$, $A \cap B = \emptyset$ and

$$\sum_{x_i \in A} x_i = \sum_{x_i \in B} x_i$$

To prove PARTITION is NP-Hard we use a reduction from SUBSET-SUM

Numbers $\{x_1, x_2, x_3 \ldots x_n, t\}$ input instance of subset sum.

Let $x = \sum_{i=1}^{n} x_i$.

If $t \le x/2$ set $x_{n+1} = x - 2t$

else $x_{n+1} = 2t - x$

$\{x_1, x_2, x_3 \ldots x_n, x_{n+1}\}$ is an instance of partition

Thus $\{x_1, x_2, \ldots x_n, t\}$ is a yes instance of subset sum iff $\{x_1, x_2, x_3, \ldots x_n, x_{n+1}\}$ is a yes instance of partition.

① SUBSET SUM ⟶ PARTITION

[YES] [YES]

$\exists$ subset of $\{x_1, x_2 \cdots x_n\}$ that sums to t

S : set of $x_i \in X$ with $\sum_{x_i \in S} x_i = t$

if $x_{n+1} = X - 2t$

$T : X \setminus S$ $\quad S' = S \cup \{x_{n+1}\}$

$\sum_{x_i \in T} x_i = X - t = t + X - 2t = X - t$

Thus for a subset S

$\exists$ partition of $\{x_1, x_2 \cdots x_{n+1}\}$

such that

$T : X \setminus S$, $S' : S \cup \{x_{n+1}\}$

$T \cup S = \{x_1, x_2 \cdots x_{n+1}\}$

② SUBSET SUM ⟵ PARTITION

[YES] [YES]

$\exists$ S, T such that

$$\sum_{x_i \in S} x_i = \sum_{x_i \in T} x_i = \frac{(X-t)+(X-t)}{2} = X - t$$

Take partition that contains x_{n+1}

and remove it ⟶ $(X-t) - (X-2t)$

$= t$

This shall be our subset whose sum turns out to be $\boxed{t}$