Quiz uploaded on DEN
under week 2 content

Basics about you

1. Your name:
2. Your e-mail address:
3. Your major and degree program:
4. Your areas of research interests (if applicable) — feel free to list multiple areas if you are undecided:
5. Titles of relevant classes you have taken before — this may include algorithms, complexity, languages & automata, graph theory, discrete mathematics, probability, linear algebra, mathematical programming, or others that you can think of:

Background Knowledge

This section tries to ascertain some basic knowledge we hope you acquired before. This is not a quiz, and your performance here will not affect your grade. However, if you have serious problems in this section, it may be in your own best interest to review the background material in order to do well in this class.

1- Which of these sorting algorithms have a worst-case running time of $\Omega(n^2)$ — mark all that apply: Bubble Sort, Heap Sort, Insertion Sort, Merge Sort, Quick Sort (with good median finding), Selection Sort.

2- Which of these sorting algorithms have a worst-case running time of $O(n \log n)$ — mark all that apply: Bubble Sort, Heap Sort, Insertion Sort, Merge Sort, Quick Sort (with good median finding), Selection Sort

3- Which of these functions are $O(n^2)$ — mark all that apply: 3, $(2n)^2$, $(\log n)^4$, 2^n, $1/100\ n^3$, log logn, $4n \log n$, $n^2 + 4n \log n$.

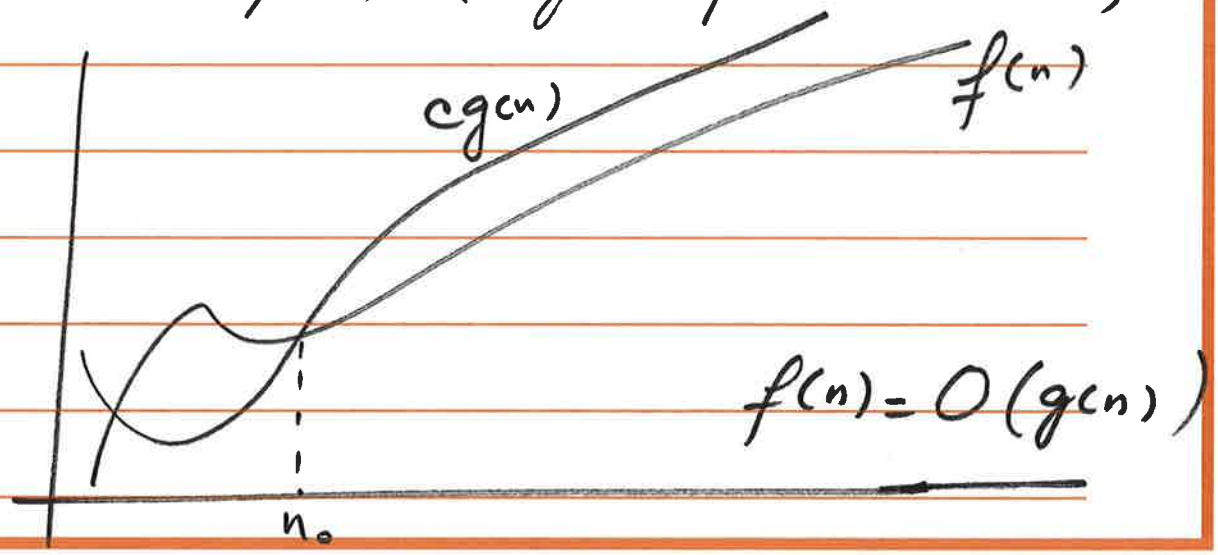
4- Which of these functions are $\Omega(n^2)$ — mark all that apply: 3, $(2n)^2$, $(\log n)^4$, 2^n, $1/100\ n^3$, log logn, $4n \log n$, $n^2 + 4n \log n$.

5- Among the following subsets of (undirected) graphs, determine which are subsets of each other: (1) cycle (2) tree, (3) forest, (4) connected graph, (5) acyclic graph, (6) bipartite graph, (7) path. For each class A, list all classes B such that the following statement holds: "every A is also a B".

6- In a graph with n nodes and m edges how long does it take to
   a. Run BFS to find out if there is a path from node A to node B
   b. Run DFS to find out if there is a path from node A to node B
   c. Run BFS to find all points that can be reached from A
   d. Run DFS to find all points that can be reached from A

7- Which of the following statements are true?
   a. BFS can be used to find the shortest path in an undirected graph with equal cost edges in linear time
   b. DFS can be used to find the shortest path in an undirected graph with equal cost edges in linear time
   c. BFS can be used to find the shortest path in a directed graph with equal cost edges in linear time
   d. DFS can be used to find the shortest path in a directed graph with equal cost edges in linear time
   e. BFS can be used to find the shortest path in a weighted undirected graph in linear time
   f. DFS can be used to find the shortest path in a weighted undirected graph in linear time
   g. BFS can be used to find the shortest path in a weighted directed graph in linear time
   h. DFS can be used to find the shortest path in a weighted directed graph in linear time
   i. BFS can be used to find the shortest path in a weighted undirected graph in quadratic time
   j. DFS can be used to find the shortest path in a weighted undirected graph in quadratic time
   k. BFS can be used to find the shortest path in a weighted directed graph in quadratic time
   l. DFS can be used to find the shortest path in a weighted directed graph in quadratic time

Formally, $O(g(n)) = \{ f(n) \mid$ there exist positive constants C and n_0 such that $0 \leq f(n) \leq c\,g(n)$ for all $n \geq n_0 \}$

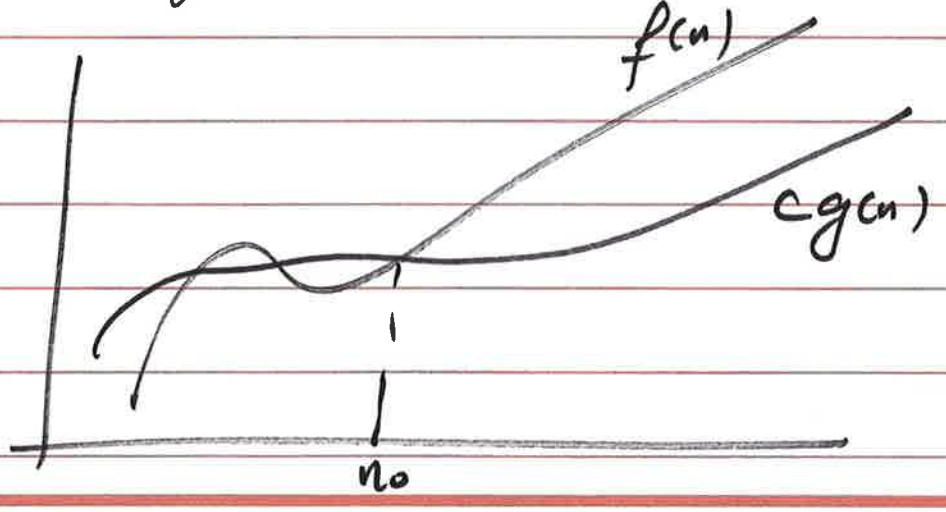


T any quadratic function is $O(n^2)$

T any linear function is $O(n^2)$

F any cubic " " $O(n^2)$

$\Omega(g(n)) = \{ f(n) \mid$ there exist positive constants C and n_0 such that $0 \leq Cg(n) \leq f(n)$ for $n \geq n_0 \}$



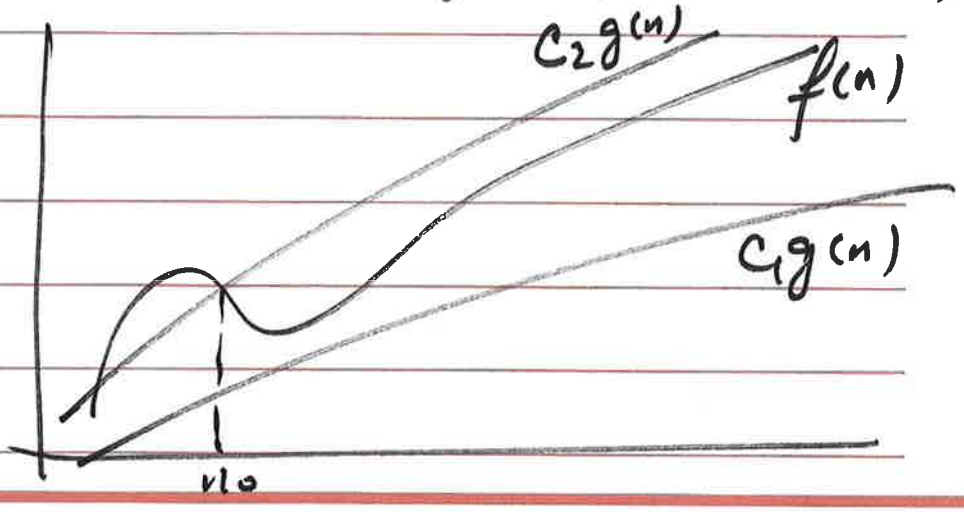
$f(n) = \Omega(g(n))$

T — any quadratic function is $\Omega(n^2)$

F — any linear " " $\Omega(n^2)$

T — any cubic " " $\Omega(n^2)$

$\Theta(g(n)) = \{ f(n) \mid$ there exist positive constants c_1, c_2, and n_0 such that $0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)$ for all $n \geq n_0 \}$



$f(n) = \Theta(g(n))$

	worst case	best case
linear search	$O(n)$	$\Omega(1)$
binary search	$O(\lg n)$	$\Omega(1)$
insertion sort	$O(n^2)$	$\Omega(n)$
merge sort	$O(n \lg n)$	$\Omega(n \lg n)$

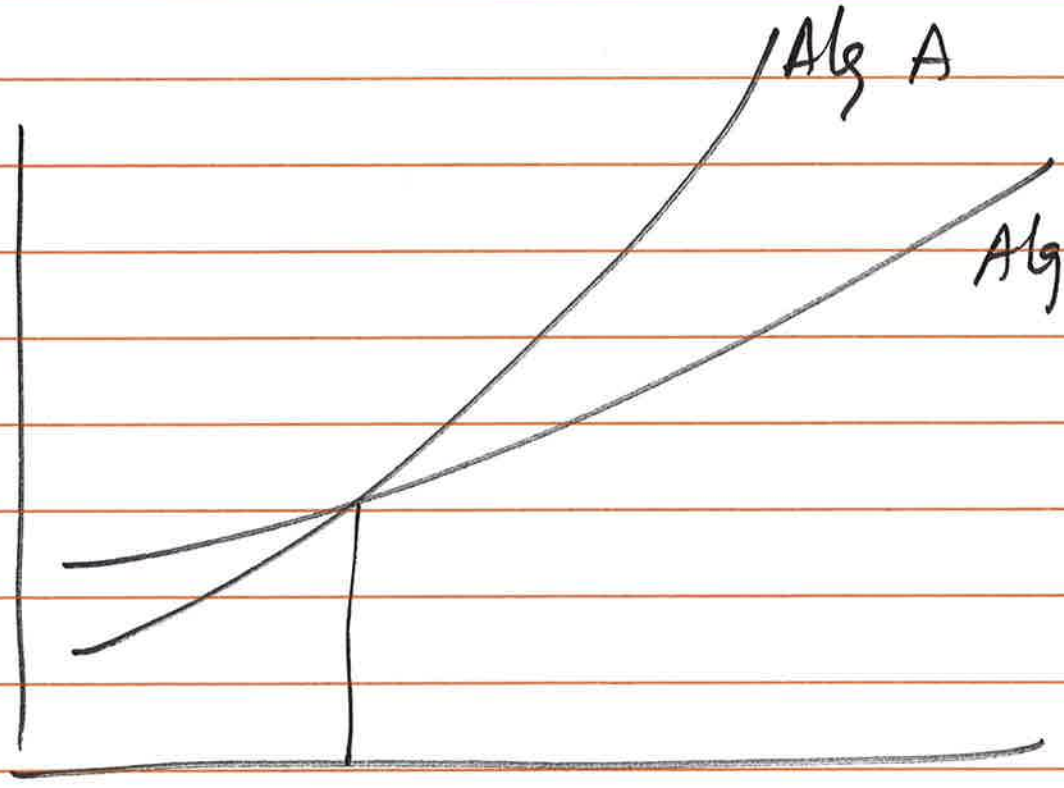
Alg A : $O(\underline{4}^{n} n^{3} \lg n)$

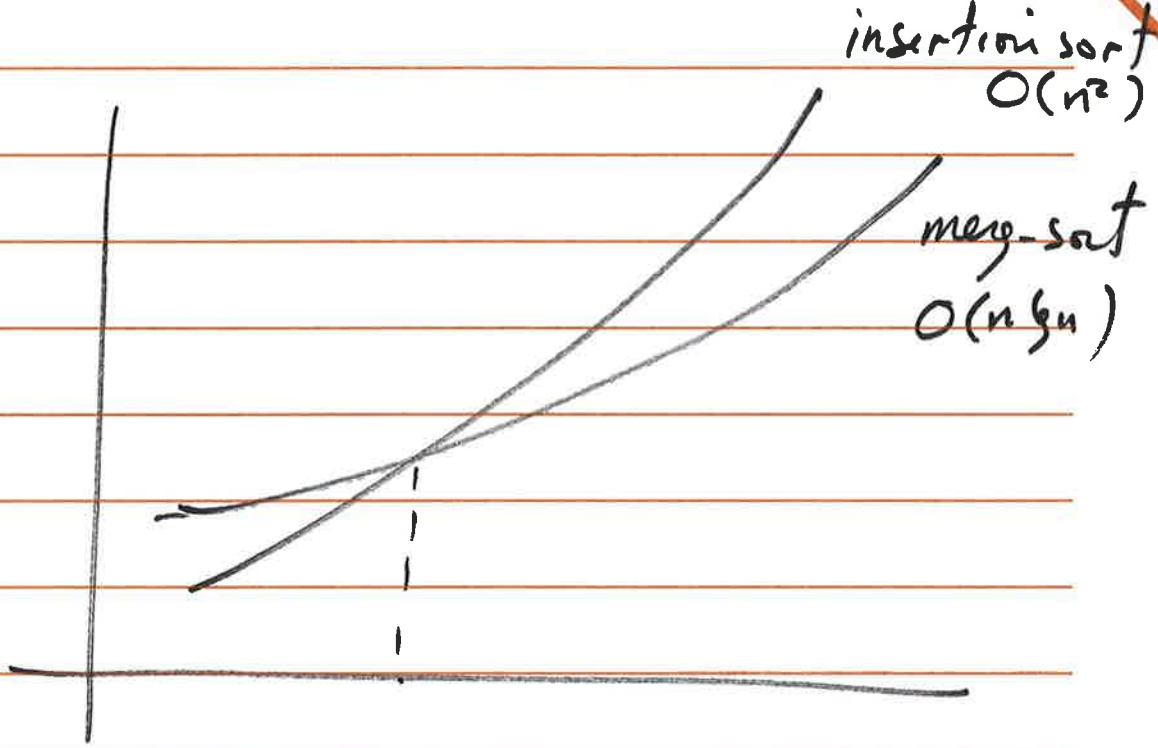
Alg B : $O(\underline{3}^{n} n^{8} (\lg n)^{2})$

1- Exponential component — fastest

2- polynomial "

3- Logarithmic " — slowest



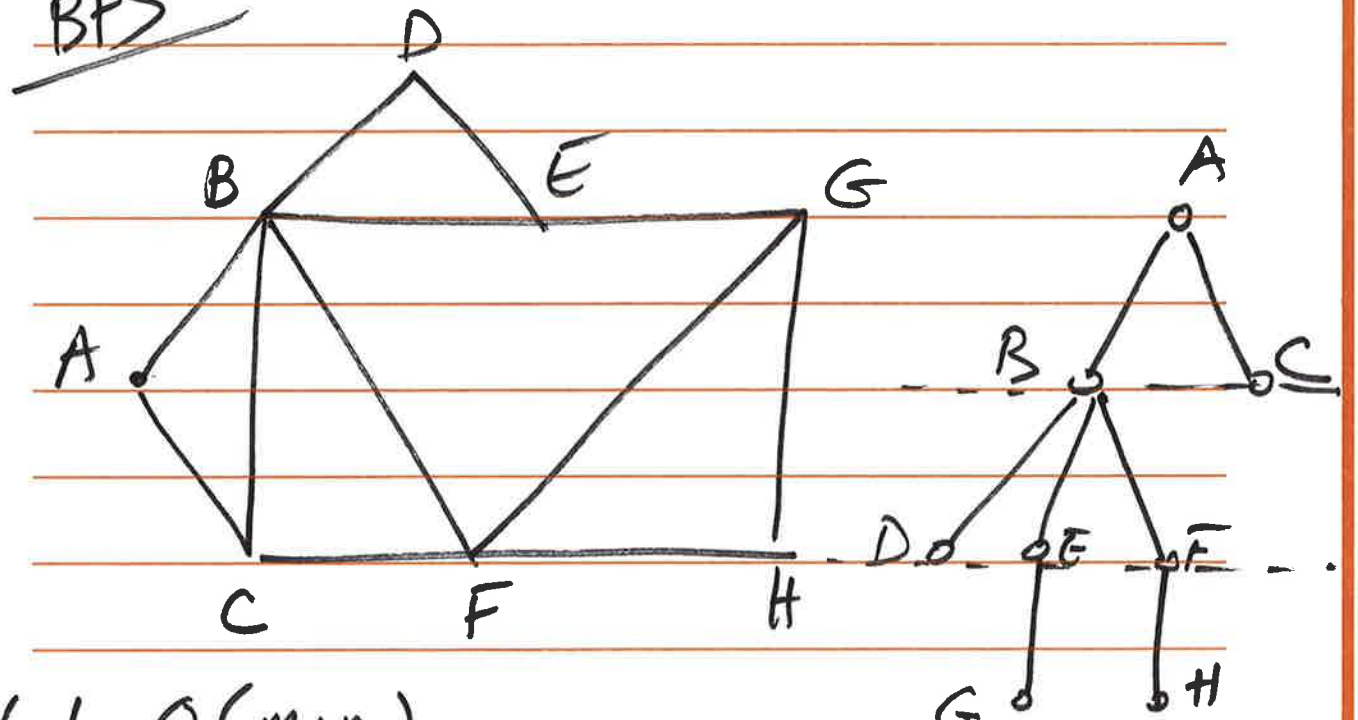


$$[A]^T [B] = [C]$$

- Find out if there is a path from A to B

- Find all nodes that can be reached from A

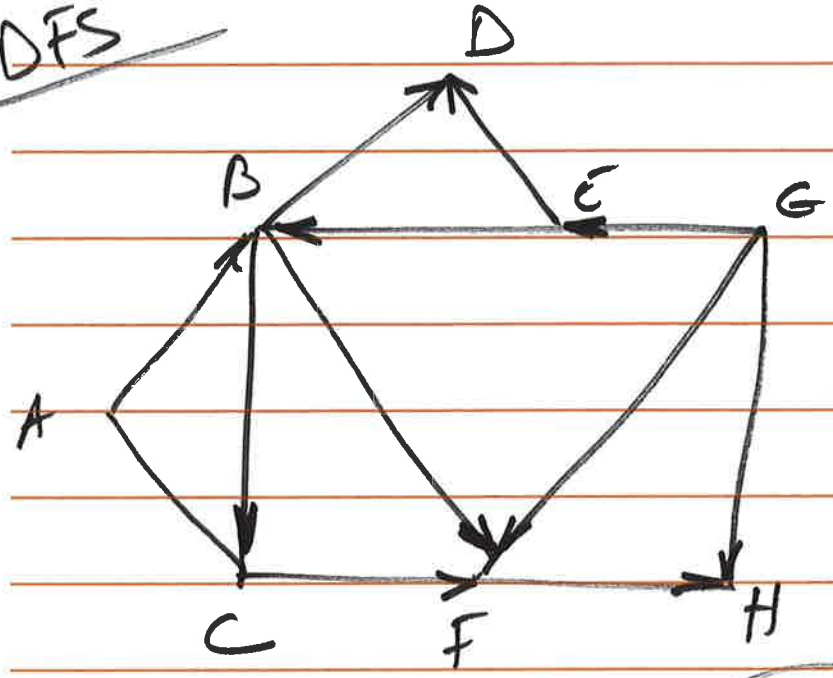
BFS



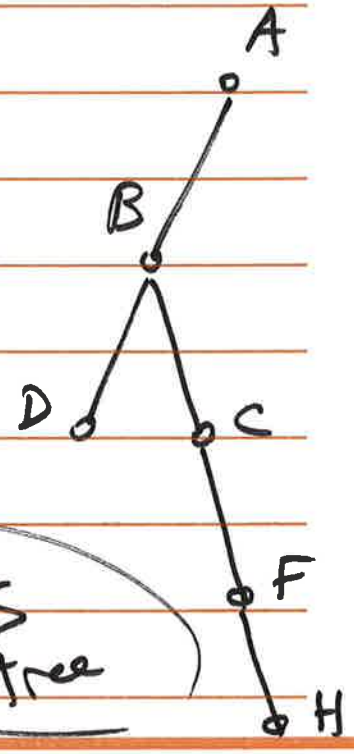
take $O(m+n)$

BFS tree

DFS

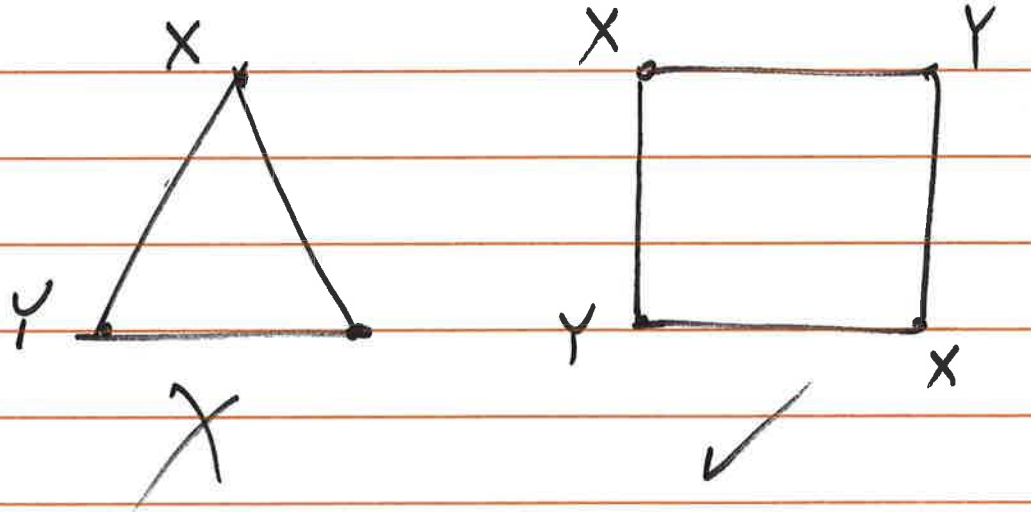


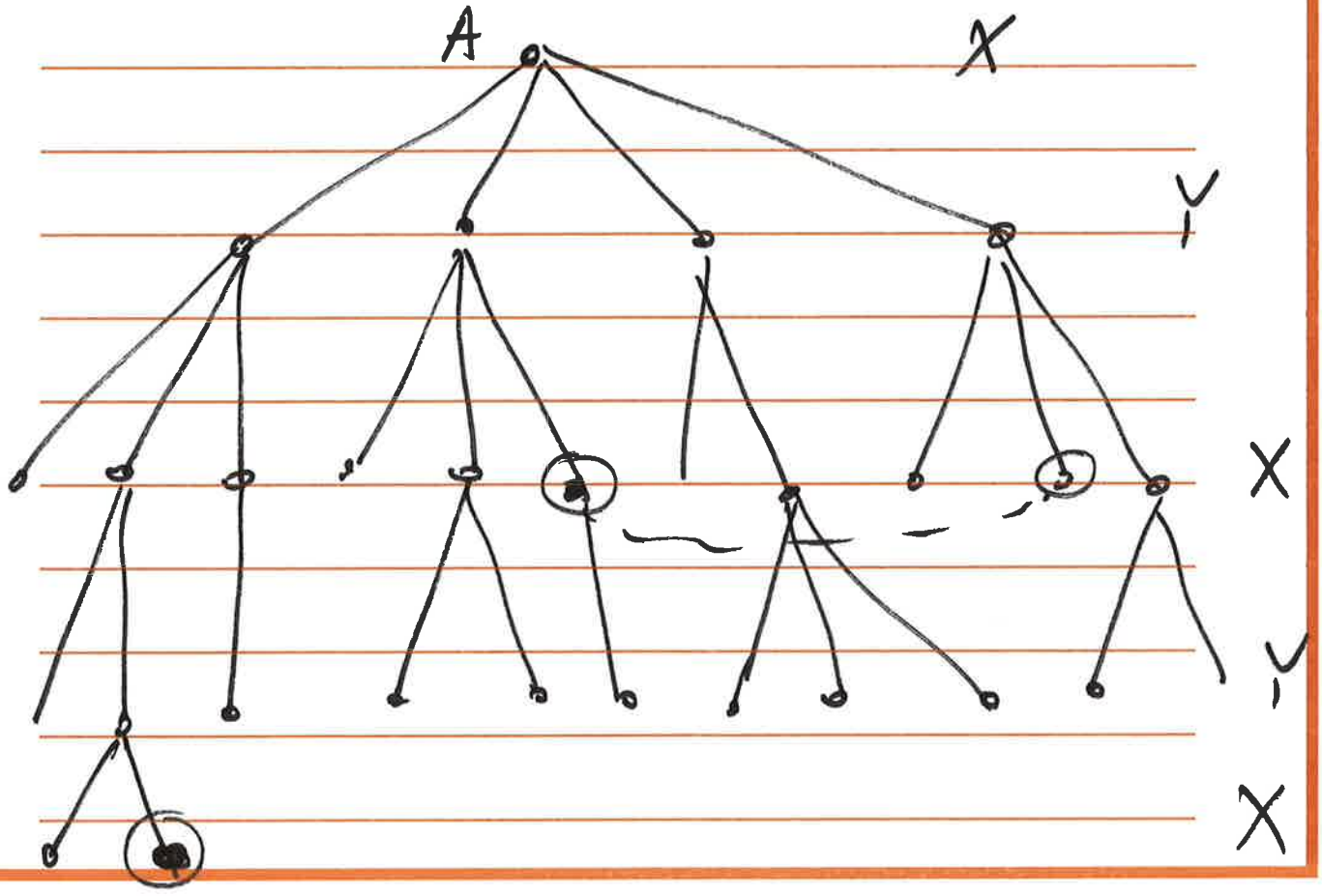
$O(m+n)$

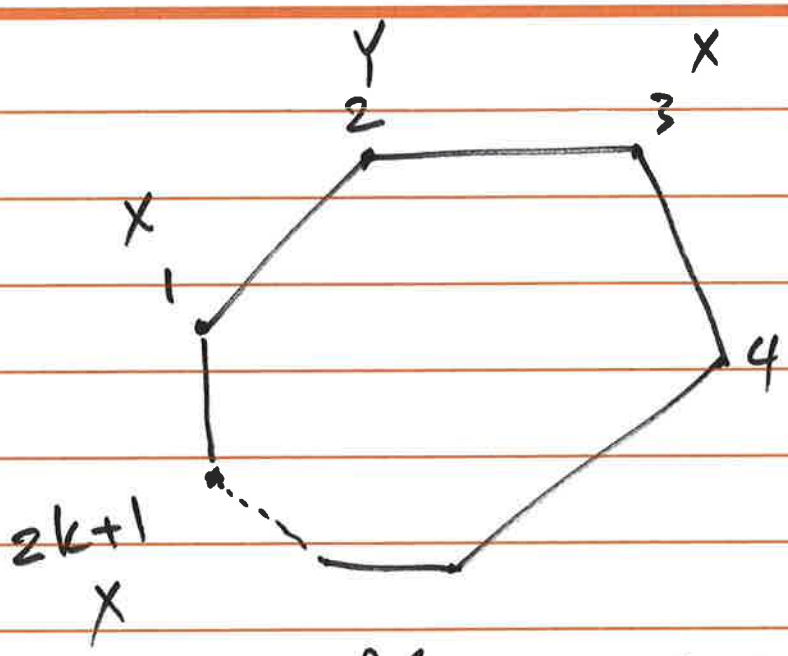


DFS tree

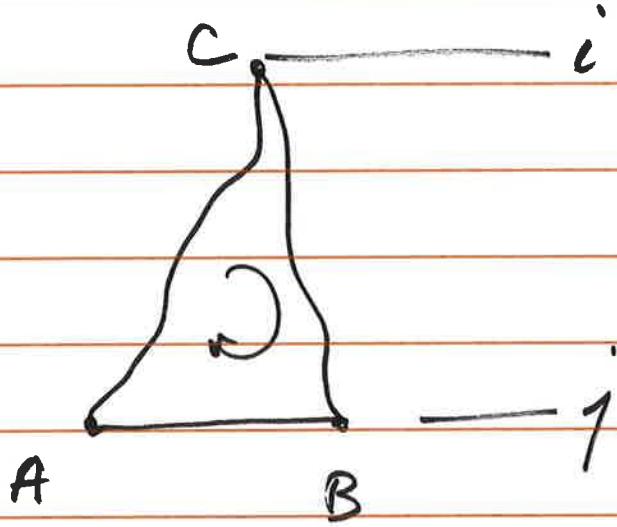
Q: How do you determine if a graph is bipartite?







FACT: If a graph G is bipartite then it cannot contain an odd cycle.



length of cycle = $2*(j-i)+1$

The alg. runs in $O(m+n)$

$O(m)$

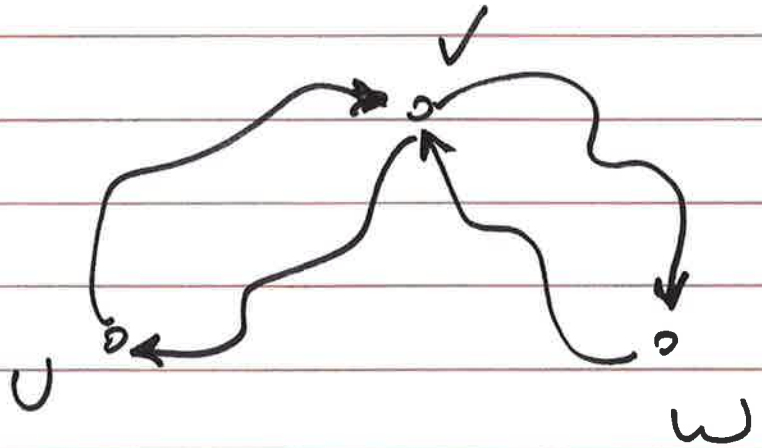
$O(m+n)$

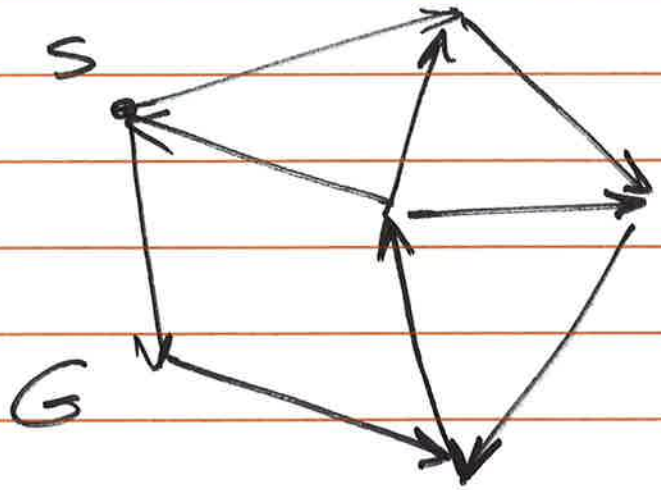
Def. A directed graph is strongly connected if there is a path from any point to any other point in the graph.

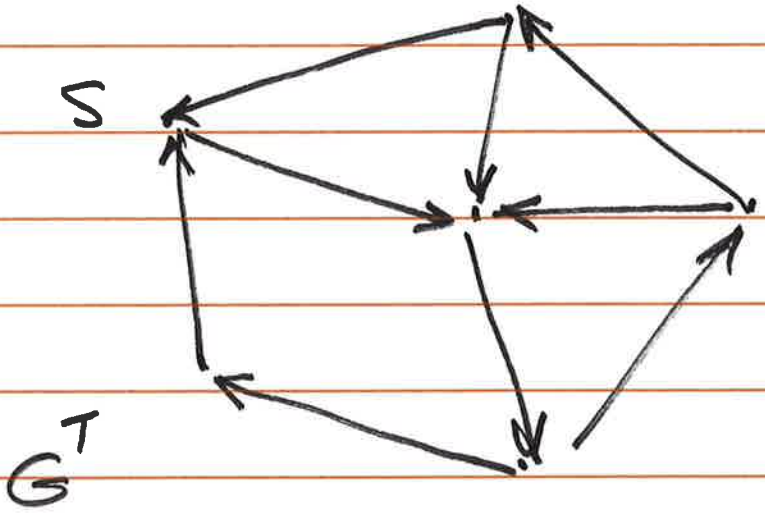
Q: How do you know if a given directed graph is strongly connected?

Brute Force: run BFS or DFS n-times.

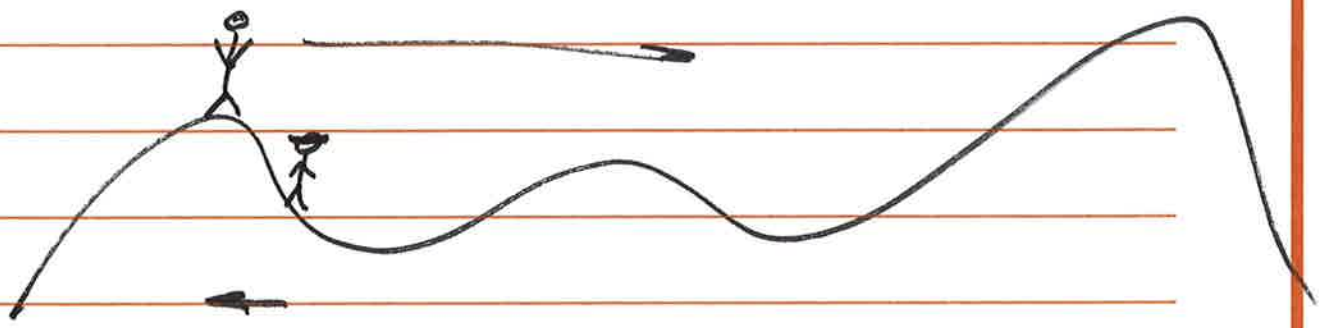
Takes $O(mn + n^2)$







Complexity is $O(m+n)$

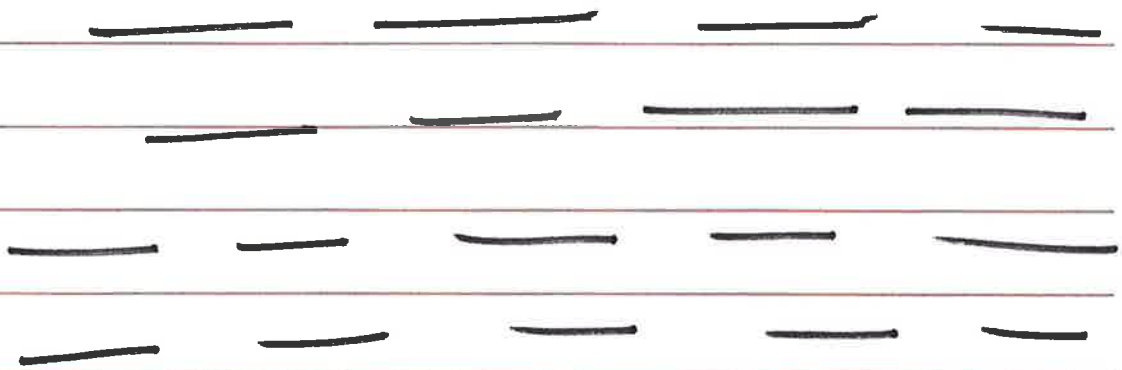


Interval scheduling Problem

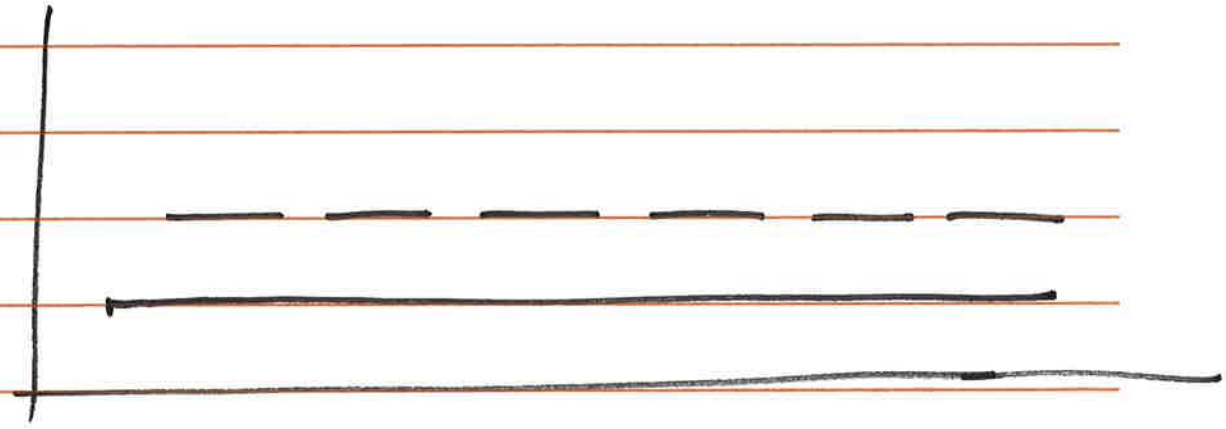
Input: Set of requests $\{1..n\}$

i^{th} request starts at $s(i)$ and ends at $f(i)$

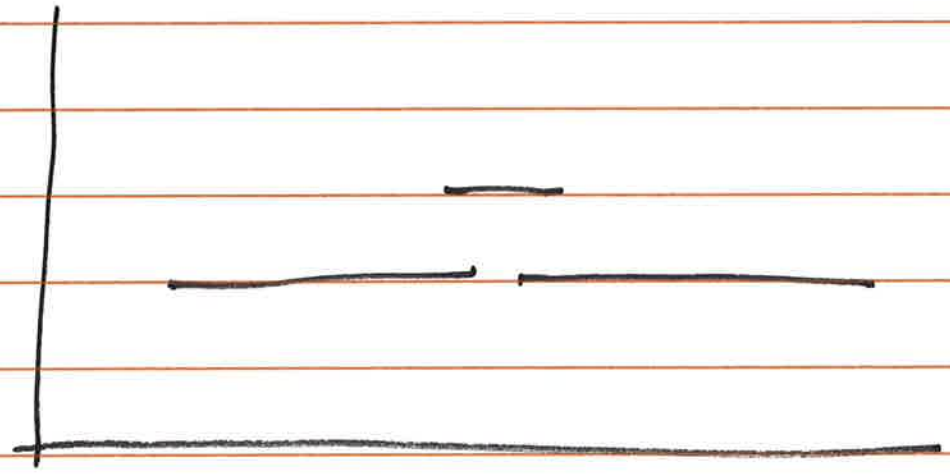
Objective: To find the largest compatible subset of these requests



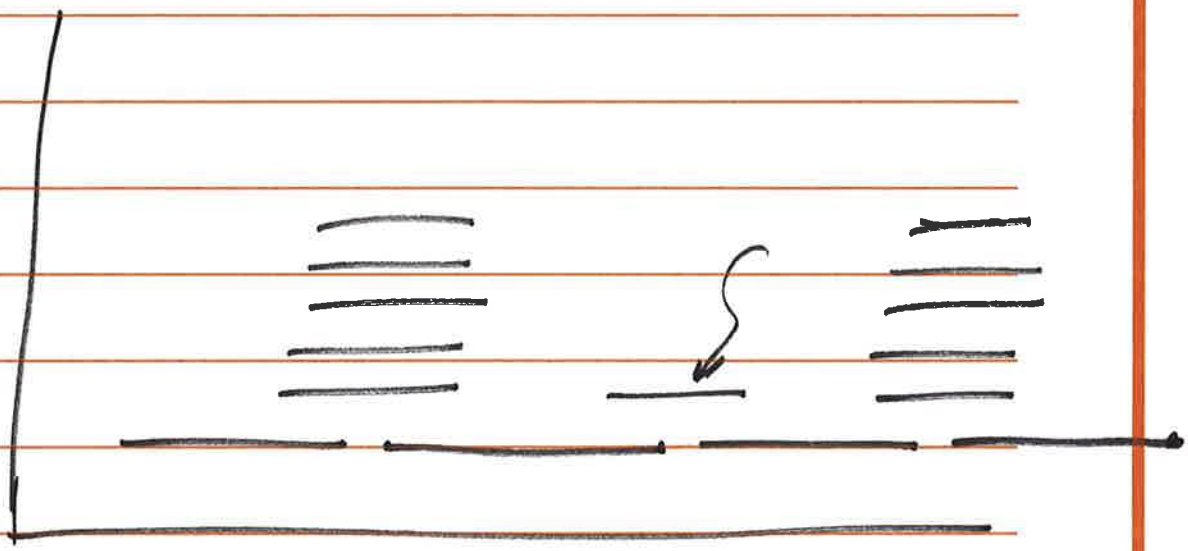
try #1 Earliest start time X



try #2 Smallest requests first X



try #3 Smallest no. of overlaps first X



try # 4 Earliest finish time first

Solution:

Initially R is the complete set of requests
& A is empty

While R is not empty

Choose a request $i \in R$ that has the smallest finish time

Add request i to A

Delete all requests from R that are not compatible w/ i

endwhile

Return A

Proof of Correctness

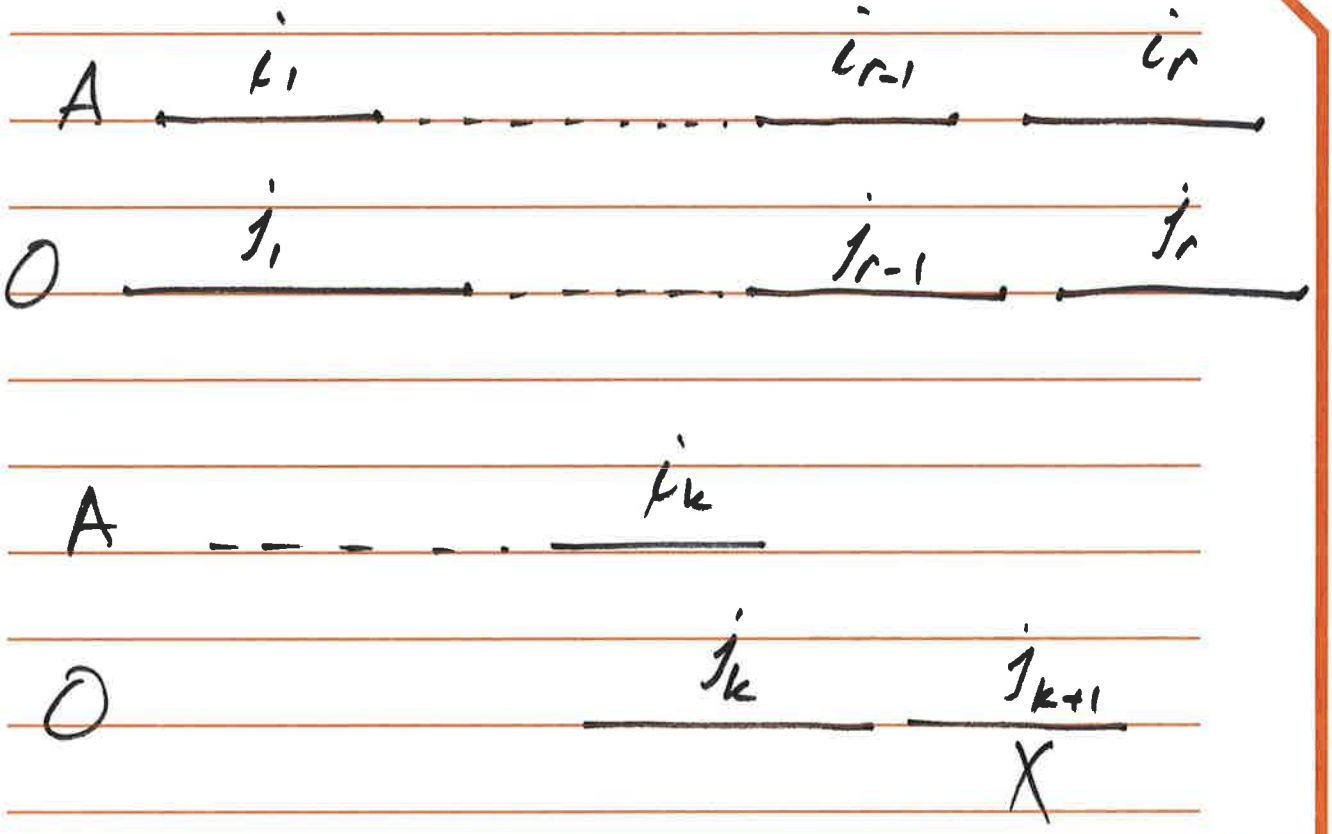
(1) Show A is a compatible set

(2) Show A is an opt. set

Say A is of size k
Say there is an opt. sol O
will prove that $|A| = |O|$

requests in A: $i_1 \ldots i_k$
" " O: $j_1 \ldots j_m$

Prove that for all indices $r \leq k$
we have $f(i_r) \leq f(j_r)$



$\Rightarrow |A| = |O|$

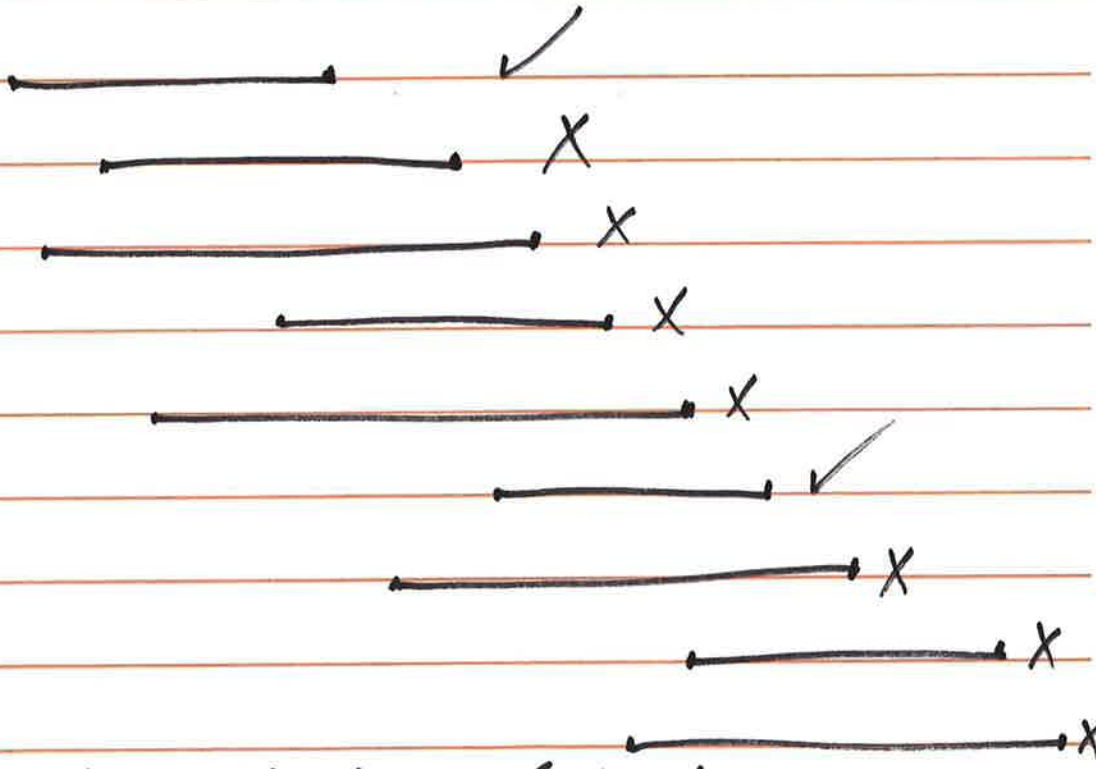
Implementation

$O(n \log n)$ — Sort requests in order of finish time and label in this order:

$f(i) \leq f(j)$ where $i < j$

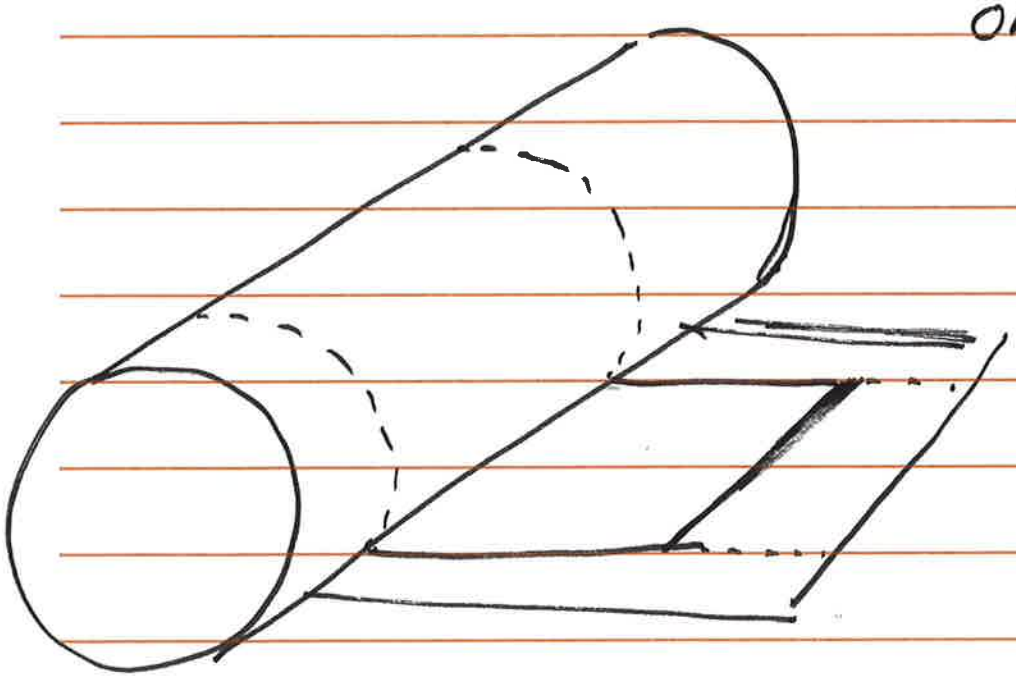
$O(n)$ — Select requests in order of increasing $f(i)$ always selecting the first

then iterate through the intervals in this order until reaching the first interval for which $s(j) \geq f(i)$

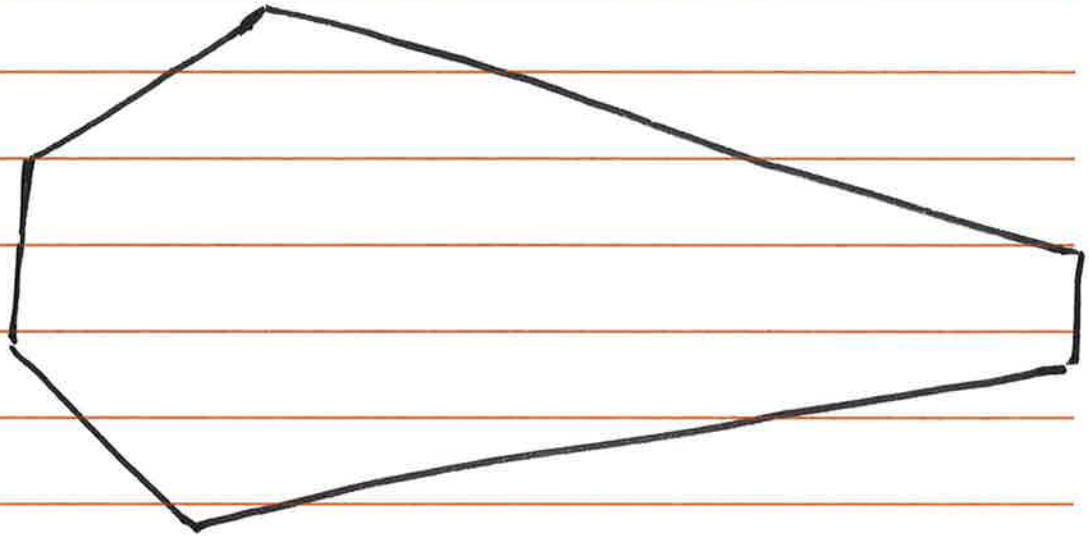


overall complexity = $O(n \log n)$

~~Vol~~ Vanilla
- Chocolate
- Strawberry



orders:
Qty
width
Grade



Fractional knapsack

Knapsack has a weight capacity of W

we are given as input a set of n objects with weight w_i and value v_i

Objective: Fill up the knapsack to its weight capacity such that the value of items in knapsack is maximized.

Ex. knapsack weight cap: 10

items	1	2	3	4	5
values	10	20	15	2	8
weights	(4)	10	5	1	2
value/weight	2.5	2	3	2	4

order: 5, 3, 1, 2, 4

(5) + (3) + 3/4 of (1)

Tot. Value $8 + 15 + 7.5 = 30.5$