peek → getFirst()

Greedy Approach

Introduction to Greedy

company HR

local optimum (os)

TIME ↑
resource ↑
↑ we take

local solution
Round1 : Online Test(1000) → local optimum → 1000
Round2 : interview (100) OS → 1000
Round3 : interview HR(20) OS → 1000

↓

10 hire → final sol (global sol)

Def : Greedy algorithms are the problem-solving Technique where we can make the locally optim-um choice at each stage & hope to achieve a global optimum(final sol)

1) optimization → Min Max → final sol
2) Sorting
3) No fixed rule
4) not realize

Binary Search, Quick Sort, Merge sort → They have fixed rules

Pros
Simple & Easy
Good enough TC

Cons
A lot of time, global optimum is not achieved

DroPack: Might not give the global optimum in all cases.

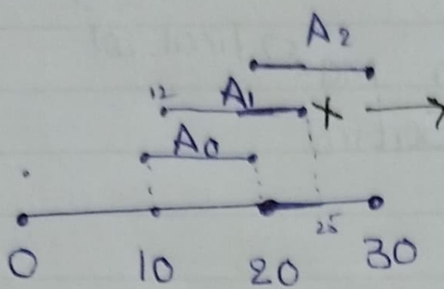
***

Activity Selection

You are given n activities with their start and end times. Select the maximum number of activities that can be performed by a single person, assuming that a person can only work on a single activity at a time. Activities are sorted according to end time.

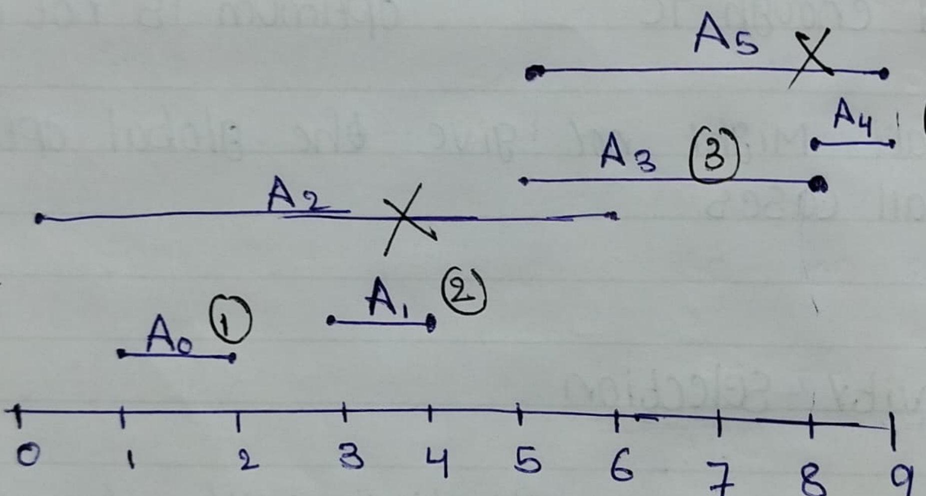
$$\begin{array}{lcl} & & A_0 \quad A_1 \quad A_2 \\ \text{Start} & = & [10,\ 12,\ 20] \\ \text{end} & = & [20,\ 25,\ 30] \end{array}$$

Ans = 2 (A_0 & A_2)



Approach

$$\begin{array}{lcl} & & A_0 \ A_1 \ A_2 \ A_3 \ A_4 \ A_5 \\ \text{Start} & = & [1,\ 3,\ 0,\ 5,\ 8,\ 5] \\ \text{end} & = & [2,\ 4,\ 6,\ 7,\ 9,\ 9] \end{array}$$



1) end basis Sort

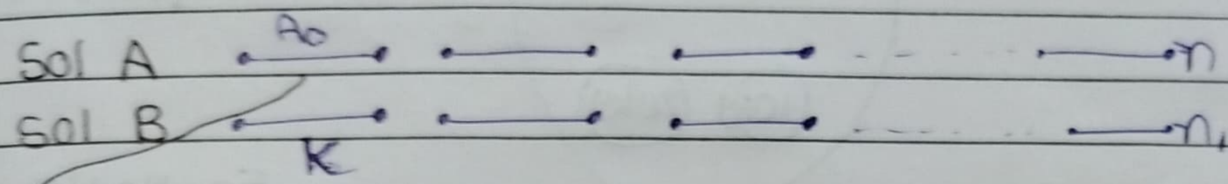
2) A_0 (is the first activity)
 ↳ end time
 → then non-overlaop disjoint

3) where Start time >= last choicen end time

Count ++;

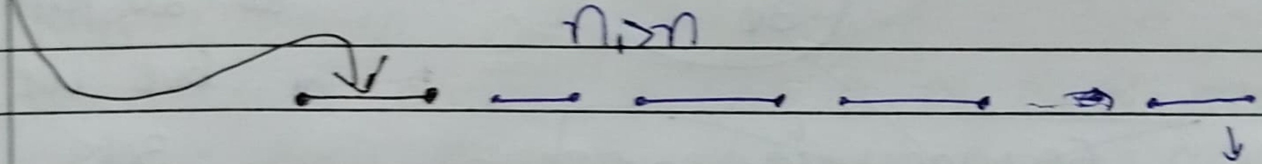
Activity Selection ** (APPit & COMMU)

MATH logic



Where Note Sol B Start with Null then second to K to n_1 (Non-overlap)

$n_1 > n$



$\boxed{n_1 = n} \rightarrow$ optimal $\quad n_1 - \cancel{1} + \cancel{1}$

Fractional knapsack

Ex:- Given the weights and values of N items, put these items in a knapsack of capacity W to get the Maximum total value in the knapsack.

Value = [60, 100, 120] $\quad$ W = 50
Weight = [10, 20, 30]

Given data

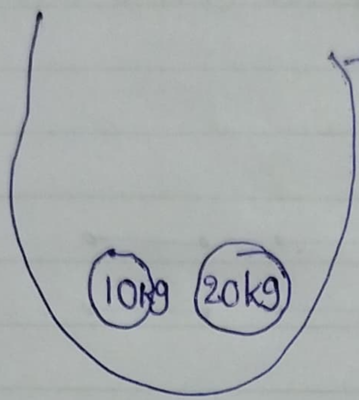
Weights ↓ Value ↑ N items

↓

Profit

W = 50

knapsack →



→ 50kg

→ Bag

$\boxed{\text{Ans:- } 240}$

50 - 20+10

50 - 30

= 20 remain

but we have

30kg

$60 + 100 + \frac{120}{30} \times 20$ → wrong

$\cancel{60 + 100 + \frac{120}{20} \times 30}$

$= 60 + 100 + \frac{120}{30} \times 20 = 60 + 100 + 80$

$= 180 + 60$

$= 240$

Code Approach

Expt. No.: Page No.: Date:

Min Absolute Difference pairs

Given two arrays A and B of equal length n. Pair each element of array A to an element in array B, such that sum S of absolute differences of all the pairs is minimum

A = [1, 2, 3] Ans : 0
B = [2, 1, 3]

-> logic

for Ex: a = 5, b = 5 (or) a = 1, b = 2

|a-b| = |5-5| = 0 |1-2| = -1 -> +1
|b-a| = |5-5| = 0 |2-1| = 1

Case1 : |1-2| + |2-1| + |3-3| = 1+1+0 = 2
Case2 : |1-3| + |2-1| + |3-2| = 2+1+1 = 4

Teacher's Signature

Case 3:- $|\cancel{2-3}\ |1-1| + |2-2| + |3-3| = 0+0+0=0$

Min

Greedy Approach

Sort A & B

Ex:- A = [1, 2, 3]

B = [2, 1, 3]

A = [1, 2, 3]

B = [1, 2, 3] then we do $|1-1| + |2-2| + |3-3|$

$= \underline{\underline{0}}$

Max Length chain of pairs

You are given n pairs of numbers. In every pair, the first number is always smaller than the second number. A pair (c, d) can come after pair (a, b) if b<c. Find the longest chain which can be formed from a given set of pairs.

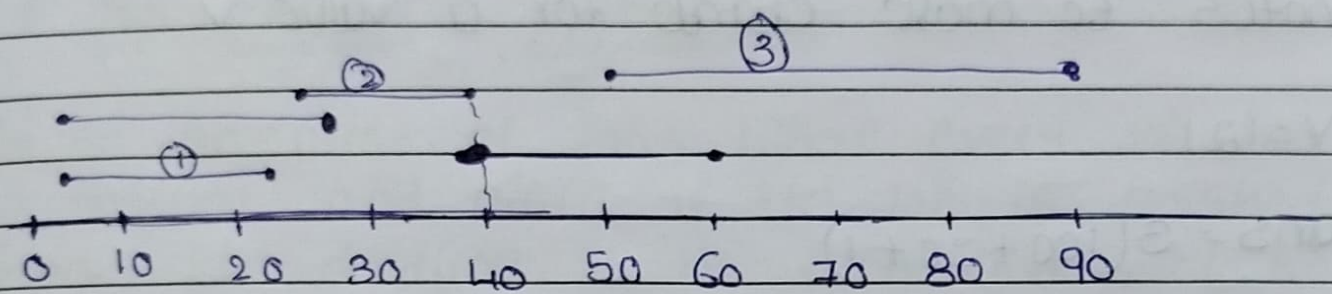
Pairs =

(5, 24)   (5, 28)   (50, 90)

(39, 60)   (27, 40)   Ans = 3

Expt. No.: .... Page No.: ....

Date: ....



Note: overlapping No

Approach

1) Sort (at last element are 24, 60, 28, 40, 90)

2) ~~Start~~ first pair

for (int i=1; i<n; i++) {

if(first > endselect->last) {

last -> update
ans++;

Teacher's Signature ____________

Indian Coins

We are given an infinite Supply of denominations [1, 2, 5, 10, 20, 50, 100, 500, 2000]. Find min no. of coins/notes to make change for a value V.

V = 121

ans = 3(100 + 20 + 1)

V = 590

ans = 4(500 + 50 + 20 + 20)

Approach

1) [1, 2, 5, 10, 20, 50, 100, - -]

1) Sort descending [200, 500, 100, 50, 20, 10, 5, 2, 1]

1 × 1 1 1

Amount = 590

590 - 500 = 90 - 50 = 40 ↓ 20 ↓ 20

count = 0

```
for (int i=0; i<n; i++) {

    if(coin[i]< amount) {
    while(corn[i] <amount) {

        count++;
```

amount = amount - coin[i];
}
}

Job Sequence problem

Given an array of jobs where every job has a deadline and profit if the job is finished before the deadline. It is also given that every job takes a single unit of time. So the minimum possible deadline for any job is 1. Maximize the total profit if only one job can be scheduled at a time.

Job A = 4, 20
Job B = 1, 10 Ans = C, A
Joc C = 1, 40
Joc D = 1, 30

① ~~A~~ time = 1, profit = 20

② B, A time = 2, profit = 10 + 20 = 30

③ ~~B~~ C, A time = 2, profit = 40 + 20 = 60 ✓

④ D, A time = 1, profit = 30 + 20 = 50

Another way of Solving this

	J_1	J_2	J_3	J_4
P:	20	10	40	30
D:	4	1	1	1

A — B — C — D

0 $\underline{J_3}$ 1 — 2 — 3 $\underline{J_1}$ 4

(hrs) 9 10 11 12 1

$J_3 + J_1 = 40+20$
$= \underline{\underline{60}}$

Approach

Step 1: Base sort (That means sort the profit's

Step 2: time = ~~0~~ ~~1~~ 2

Step 3:

Ans (array): | 1 | 4 |

```
for (int i=0; i<jobs; i++)
{
if( job(deadline) > time) {
~~add ans~~ add
time++
} }
```