HEAP ...

Video-16

(Priority Queue)

Leetcode -857

Hard

Facebook
Instagram → codestorywithMIK

Twitter → cswithMIK

→ codestorywithMIK

Company :- Google

857. Minimum Cost to Hire K Workers

Hard | Topics | Companies

There are `n` workers. You are given two integer arrays `quality` and `wage` where `quality[i]` is the quality of the `i^th` worker and `wage[i]` is the minimum wage expectation for the `i^th` worker.

We want to hire exactly `k` workers to form a **paid group**. To hire a group of `k` workers, we must pay them according to the following rules:

1. Every worker in the paid group must be paid at least their minimum wage expectation.
2. In the group, each worker's pay must be directly proportional to their quality. This means if a worker's quality is double that of another worker in the group, then they must be paid twice as much as the other worker.

Given the integer `k`, return *the least amount of money needed to form a paid group satisfying the above conditions*. Answers within `10^-5` of the actual answer will be accepted.

Example :- quality = { 10, 20, 5 }

K = 2

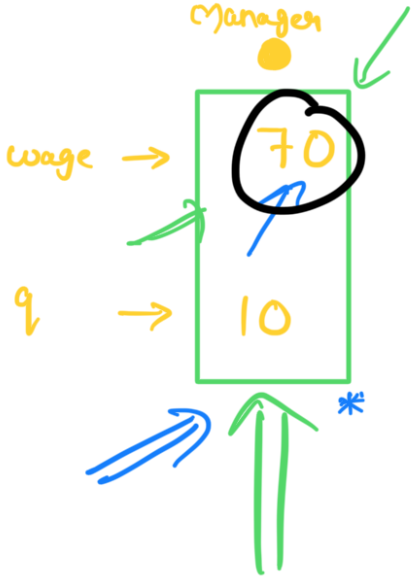
min_wage = { 70, 50, 30 }

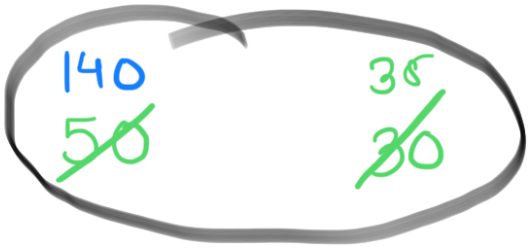
0 1

Output :- 105.000

K = { 70, 35 }

= 105





20 5

Finding K min. wages

Heap. (max)

70 + 35 = 105

$\frac{70}{x} = \frac{10}{5}$

$x = 35$

min_wage [manager] quality [manager]

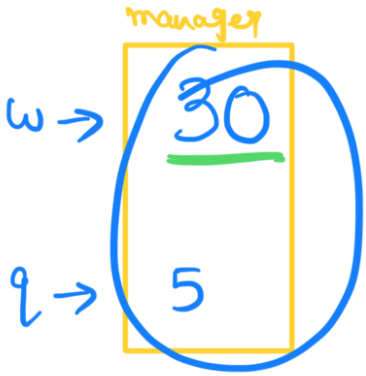
min-wage [worker] — quality [worker]

ration_manager

=> min-wage [manager] * quality [worker] = min-wage[worker]

quality [manager]

$\left(\frac{70}{10}\right) * 5 = 35$



120 → 50 :)

20

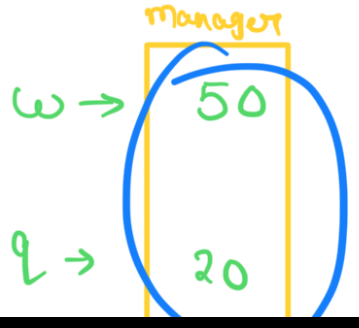
60 → 70

10

K=(30, 120) = 150

30 + 120 = 150

$\left(\frac{30}{5}\right) * 10 = 60$



25 → 70 :)

10

12.5 → 30

5

$\left(\frac{50}{20}\right) * 5 = \frac{25}{2}$

50+ ✗

Story to code:-

1. for (manager = 0 ; manager < n ; manager++)

2. manager_ratio = min_wage [manager] / quality[manager];

vector <double> group ;

3. for (worker = 0 ; worker < n ; worker++) {

worker_wage = manager_ratio * quality[worker];

if (worker_wage >= min_wage [worker]){

}

4. if (group.size() < K) continue ;

5. Sort → K min wages.

Heap → K min wages

sum-wages ;

result = min(result, sum-wages);

6. return result;

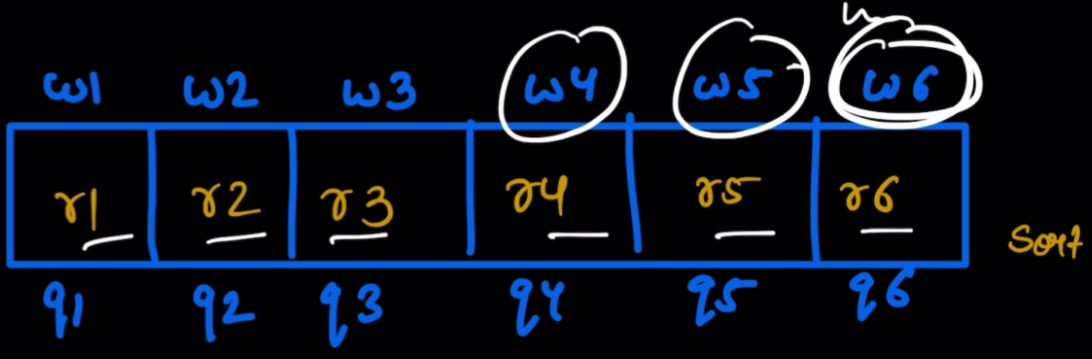
Better Brute Force

worker-wage >= min-wage [worker]

manager-ratio * quality[worker] >= min-wage[worker]

$$\text{manager-ratio} >= \frac{\text{min-wage [worker]}}{\text{quality [worker]}}$$

manager-ratio >= worker-ratio



K = 4

{10, 20, 5} → {(7,10), (2.5,20), (6,5)}

{70, 50, 30} {(2.5,20), (6,5), (7,10)} , K = 2

1. Ratio of workers.

2. Sort ascending.

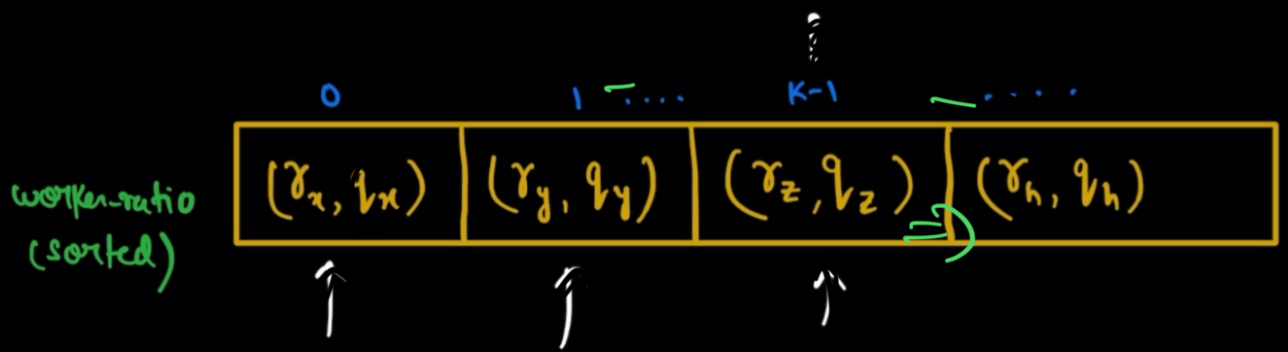
3. for (manager = K-1 ; man < n ; man++) {

```
manRatio = workerRatio[K-1].first;
for(worker=0; worker<=manager; worker++){
    worker-wage = manRation * workeRatio[worker].second;
    group.push_back(worker-wage);
}
```

4. Keep updating result.

5. At last return the best result.

Optimal Approach



(•) i^{th} manager → managerRatio (mR)

$$(mR * q_x) + (mR * q_y) + (mR * q_z)$$
$$= mR\,(q_x + q_y + q_z)$$

(•) $i+1^{th}$ manger → managRatio (mR)

$mR *$

$mR'\,(q_x + q_y + q_z + q_h)$

Sum =

K = 3

Complete Dry Run ...

q = {10, 20, 5}

K = 2

w = {70, 50, 30}

workerRatio = {(7, 10), (2.5, 20), (6, 5)}

= {(2.5, 20), (6, 5), (7, 10)}, K = 2

0 2

mR = 7

q_sum = (20 + 5 + 10) - 20

= 15

result = 7 * 15

= 105

result = 150

105