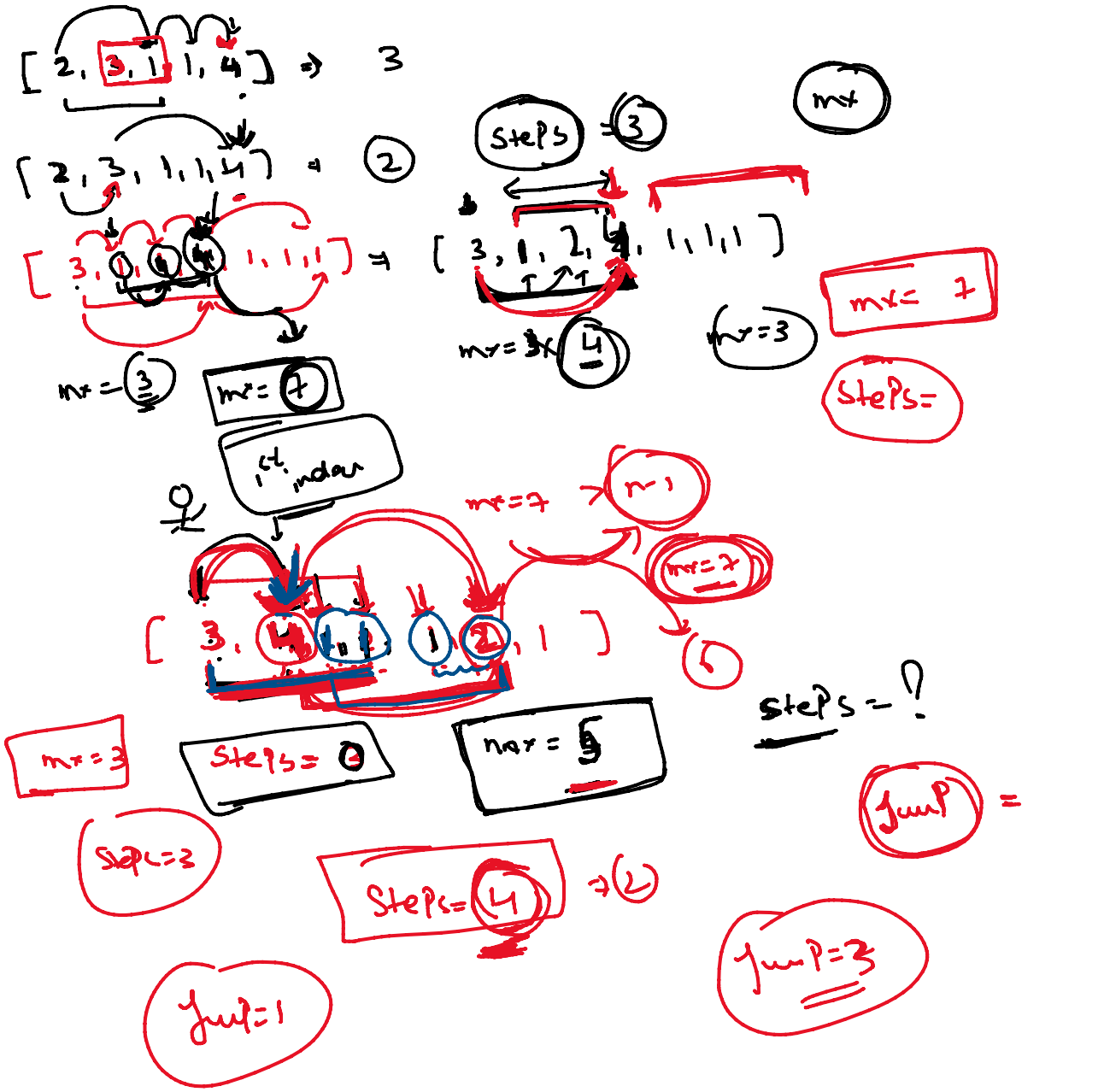
CLASS-65

Greedy

Jump Game 1

Jump Game 2



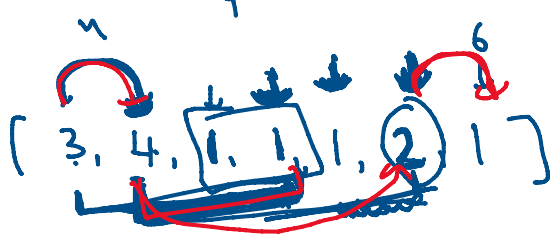

```
int jump(vector<int>& nums)
{
    n = nums.size();
```

```
Y   n = nums.size();
    if (n <= 1) return 0;
    if (nums[0] >= n-1) return 1;

    int max_reach = nums[0];
    int steps = nums[0];
    int jump = 1;

    for (i = 1; i < n; i++)
    {   if (i == n-1)
            return jump;
        if (nums[i] + i >= n-1)
            return jump + 1;
        max_reach = max(max_reach, nums[i] + i);
        step--;
        if (step == 0)
        {   jump++;
            if (i > max_reach) return -1;
            steps = max_reach - i;
```

Y



mx = 5 → mx = 7

mx = 5 → 7

steps = 0

jump = 2 + 1 = 3

[1, 2, [10, 20], [0, 100, 200, 500 - -]]

Greedy Approach

2
419
×13
12 f7
9 x
419
× 14

186, 419, 83, 408 = 6249
- 5866
383
372

Greedy, Recursion

Greedy Problem

[10, 20] [30, 200] [400, 50] [30, 20]

a, b
n n

10 + 30 + 20 = 110

[10, 20] [30, 10] [30, 200] [400, 50]

410 + 220
= 630

[10, 20] [300, 20]

Sort diff

clear

500, 1,00,000

99,000

300, 310

50

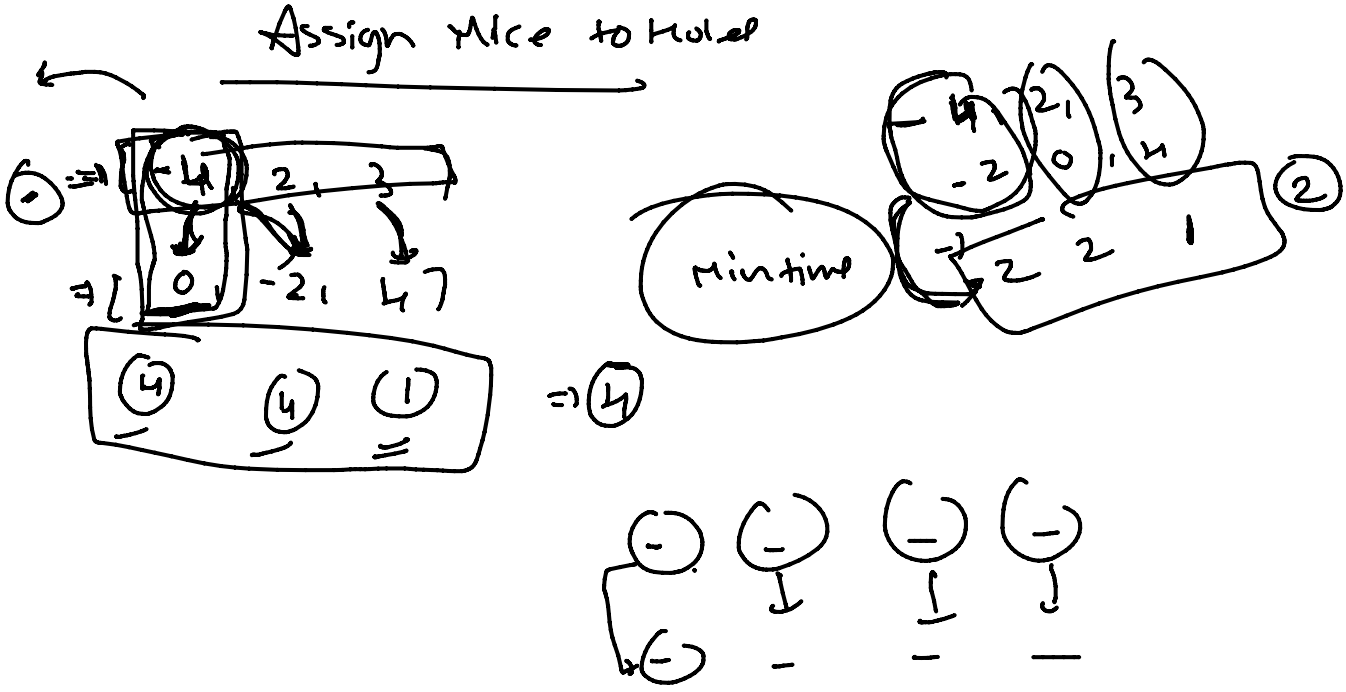
a
n

b
n

bool cmp(vector<int> a, vector<int> b)
{ return abs(a[0] - a[1]) >= abs(b[0]-b[1]) }

```
sort(costs.begin(), costs.end(), cmp);
int n= cost.size();
int a=0;
int b=0;
int temp= n/2;
int cost=0;
For (i=0; i<n; i++)
{ int mn= min(costs[i][0], costs[i][1]);
  if(mn== cost[i][0])
  { if(a<temp)
    { a++;
      cost += costs[i][0];
    else
    { b++, cost += costs[i][1];
  else {
    if(b<temp)
    { b++;
      cost += costs[i][1];
    else
    { a++; cost += costs[i][0];
  }
}
return cost;
```

(259, 770) (800, 100)




```
Sort (a.begin(), a.end()) //Mice
Sort (b.begin(), b.end()) // holes
int n= a.size();
int time = INT_MIN;
for (int i=0; i<n; i++)
{     time = max( time, abs (a[i] - b[i]) ),
}
return time;
```