Walchand College of Engineering, Sangli
Computer Science & Engineering
Third Year
Course: Design and analysis of algorithm Lab
Lab course coordinator:
Dr. B. F. Momin- Batch: - T6, T7, T8
Mr. Kiran P. Kamble- Batch: - T1, T2, T3, T4, T5

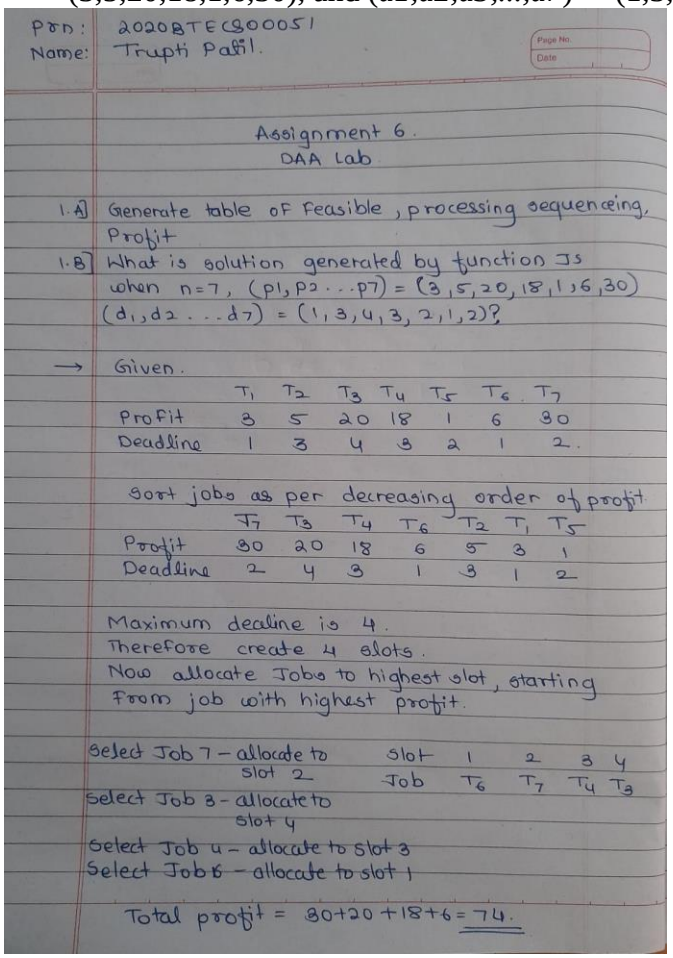
Week 6 Assignment

Greedy method

Name: Trupti Rajendra Patil

Prn: 2020BTECS00051

To apply Greedy method to solve problems of
1) Job sequencing with deadlines
1.A) Generate table of feasible,proceesing sequencing , profit .
1.B) What is the solution generated by the function JS when n=7, (*p1,p2,...,p7*) = (3,5,20,18,1,6,30), and (*d1,d2,d3,...,d7*) = (1,3,4,3,2,1,2)?



1.C) **Input**: Five Jobs with following deadlines and profits

JobID	Deadline	Profit
a	2	100
b	1	19
c	2	27
d	1	25
e	3	15

Output: Following is maximum profit sequence of jobs:
c, a, e

1.D) Study and implement Disjoint set algorithm to reduce time complexity of JS from O(n^2) to nearly O(n)

Brute force:
Algorithm:

a. Sort all jobs in decreasing order of profit.
b. Iterate on jobs in decreasing order of profit.For each job , do the following :
    i. Find a time slot i, such that slot is empty and i < deadline and i is greatest.Put the job in this slot and mark this slot filled.
    ii. If no such i exists, then ignore the job.

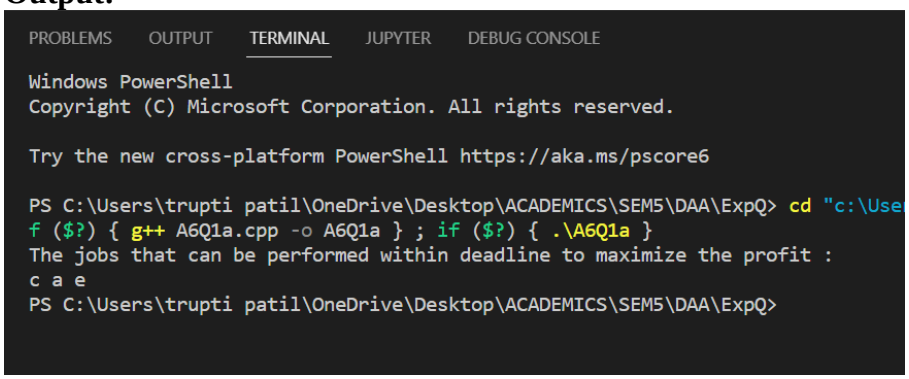
Code:

```
     #include<bits/stdc++.h>

using namespace std;

struct job{
    char jobId;
    int deadline;
    int profit;
};

bool compare(job a,job b){
    return (a.profit > b.profit);
}

int main(){
    job jobs[] =
{{'a',2,100},{'b',1,19},{'c',2,27},{'d',1,25},{'e',3,15}};
    int n=sizeof(jobs)/sizeof(jobs[0]);
```

```
    // sort according to increasing order of
    sort(jobs,jobs+n,compare);

    int result[n]={0};
    bool slot[n]={false};

    for(int i=0;i<n;i++){
        for(int j=min(jobs[i].deadline,n)-1;j>=0;j--){
            if(slot[j]==false){
                result[j]=i;
                slot[j]=true;
                break;
            }
        }
    }

    cout<<"The jobs that can be performed within deadline
to maximize the profit :"<<endl;
    for(int i=0;i<n;i++){
        if(slot[i]){
            cout<<jobs[result[i]].jobId<< " ";}
    }

}
```

Complexity Analysis:
Time complexity: O(n^2)
Space Complexity: O(n)

Output:



Disjoint set algorithm:
Algorithm:

1. Sort all jobs in decreasing order of profit.

2. Initialize the result sequence as first job in sorted jobs.
3. Do following for remaining n-1 jobs
   If the current job can fit in the current result sequence without missing the deadline, add current job to the result. Else ignore the current job.

Code:

```
     #include<bits/stdc++.h>

using namespace std;

struct Job{
    char id;
    int deadLine, profit;
};

struct DisjointSet{
    int *parent;

    DisjointSet(int n){
        parent = new int[n+1];

        for (int i = 0; i <= n; i++)
            parent[i] = i;
    }

    int find(int s){
        if (s == parent[s])
            return s;
        return parent[s] = find(parent[s]);
    }

    void merge(int u, int v){
        parent[v] = u;
    }
};

bool cmp(Job a, Job b){
```

```
    return (a.profit > b.profit);
}

int main()
{
    Job arr[] = { { 'a', 2, 100 }, { 'b', 1, 19 },{ 'c',
2, 27 }, { 'd', 1, 25 },{ 'e', 3, 15 } };
    int n = sizeof(arr) / sizeof(arr[0]);

    sort(arr, arr + n, cmp);

    int maxDeadline = INT_MIN;
    for (int i = 0; i < n; i++){
        maxDeadline = max(maxDeadline, arr[i].deadLine);
    }

    DisjointSet ds(maxDeadline);

    cout<<"The jobs that can be performed within deadline
to maximize the profit :"<<endl;
    for (int i = 0; i < n; i++){
        int availableSlot = ds.find(arr[i].deadLine);

        if (availableSlot > 0){
            ds.merge(ds.find(availableSlot -
1),availableSlot);
            cout << arr[i].id << " ";
        }
    }
    return 0;
}
```

Complexity Analysis:
Time complexity: O(nlogd)
n - total number of jobs
d - maximum possible deadline.
Space Complexity: O(d)

Ouput:

```
PROBLEMS  OUTPUT  TERMINAL  JUPYTER  DEBUG CONSOLE

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\trupti patil\OneDrive\Desktop\ACADEMICS\SEM5\DAA\ExpQ> cd "c:\Use
f ($?) { g++ A6Q1b.cpp -o A6Q1b } ; if ($?) { .\A6Q1b }
The jobs that can be performed within deadline to maximize the profit :
a c e
PS C:\Users\trupti patil\OneDrive\Desktop\ACADEMICS\SEM5\DAA\ExpQ>
```

2) To implement Fractional Knapsack problem 3 objects (n=3).

(w1,w2,w3) = (18,15,10)
(p1,p2,p3) = (25,24,15)
M=20
With strategy

a) Largest-profit strategy

Algorithm:

1. Sort the vector according to decreasing order of profit
2. Traverse the sorted vector till capacity of knapsack is not full
3. For every iteration check if the weight of current object is less than or equal to knapsack capacity, if yes then include it.
4. Else take the fractional part according to need.

Code:

```cpp
#include<bits/stdc++.h>
using namespace std;

int main(){
    // {profit,weight}
    vector<pair<int,int>> v={{25,18},{24,15},{15,10}};
    int capacity=20;
    float ans=0;

    // Sort according to decreasing order of profit
    sort(v.begin(),v.end(),greater<>());

    int i=0;
    while(capacity!=0){
        // if capacity of knapsack is more than or
equal to current weight then add it in knapsack
        if(capacity>=v[i].second){
            ans+=v[i].first;
```

```
            capacity-=v[i].second;
        }else{ //take the fractional part of it
            float x=(float)v[i].first/v[i].second;
            float y=x*capacity;
            ans+=y;
            capacity-=x;
        }
        i++;
    }

    cout<<"The maximum profit is "<<ans<<endl;
}
```

Complexity Analysis:
Time Complexity: O(nlogn)
Space Complexity: O(1)

Output:

```
PROBLEMS  OUTPUT  TERMINAL  JUPYTER  DEBUG CONSOLE

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\trupti patil\OneDrive\Desktop\ACADEMICS\SEM5\DAA\ExpQ> cd "c:\Users\tr
f ($?) { g++ A6Q2a.cpp -o A6Q2a } ; if ($?) { .\A6Q2a }
The maximum profit is 28.2
PS C:\Users\trupti patil\OneDrive\Desktop\ACADEMICS\SEM5\DAA\ExpQ>
```

b) Smallest-weight strategy

Algorithm:

1. Sort the vector according to increasing order of weight
2. Traverse the sorted vector till capacity of knapsack is not full
3. For every iteration check if the weight of current object is less than or equal to knapsack capacity, if yes then include it.
4. Else take the fractional part according to need.

Code:

```
#include<bits/stdc++.h>
using namespace std;

int main(){
    // {weight,profit}
    vector<pair<int,int>> v={{18,25},{15,24},{10,15}};
```

```
    int capacity=20;
    float ans=0;

    // Sort according to increasing of weight
    sort(v.begin(),v.end());

    int i=0;
    while(capacity!=0){
        // if capacity of knapsack is more than or
equal to current weight then add it in knapsack
        if(capacity>=v[i].first){
            ans+=v[i].second;
            capacity-=v[i].first;
            // cout<<ans<<" "<<capacity<<endl;
        }else{ // take the fractional part of it
            int c=capacity;
            float x=(float)capacity/v[i].first;
            float y=x*v[i].second;

            ans+=y;
            capacity-=c;
        }
        i++;
    }

    cout<<"The maximum profit is "<<ans<<endl;
}
```

Complexity Analysis:
Time Complexity: O(nlogn)
Space Complexity: O(1)

Output:

```
PROBLEMS  OUTPUT  TERMINAL  JUPYTER  DEBUG CONSOLE

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\trupti patil\OneDrive\Desktop\ACADEMICS\SEM5\DAA\ExpQ> cd "c:\Users\trupti pat
f ($?) { g++ A6Q2b.cpp -o A6Q2b } ; if ($?) { .\A6Q2b }
The maximum profit is 31
PS C:\Users\trupti patil\OneDrive\Desktop\ACADEMICS\SEM5\DAA\ExpQ>
```

c) Largest profit-weight ratio strategy

Algorithm:

1. Sort the vector according to decreasing order of profit/weight ratio
2. Traverse the sorted vector till capacity of knapsack is not full
3. For every iteration check if the weight of current object is less than or equal to knapsack capacity, if yes then include it.
4. Else take the fractional part according to need.

Code:

```
#include<bits/stdc++.h>
using namespace std;

bool sorted(pair<int,int>& a,pair<int,int>& b){
    float d1=(float)a.first/a.second;
    float d2=(float)b.first/b.second;

    if(d1 > d2){
        return true;
    }
    return false;
}

int main(){
    // {profit,weight}
    vector<pair<int,int>> v={{25,18},{24,15},{15,10}};
    int capacity=20;
    float ans=0;

    // Sort according to decreasing order of
profit/weight ratio
    sort(v.begin(),v.end(),sorted);

    int i=0;
    while(capacity!=0){
        // if capacity of knapsack is more than or
equal to current weight then add it in knapsack
        if(capacity>=v[i].second){
            ans+=v[i].first;
            capacity-=v[i].second;
        }else{ //take fractional part of it
```

```
            int c=capacity;
            float x=(float)capacity/v[i].second;
            float y=x*v[i].first;

            ans+=y;
            capacity-=c;
        }
        i++;
    }

    cout<<"The maximum profit is "<<ans<<endl;
}
```

Complexity Analysis:
Time Complexity: O(nlogn)
Space Complexity: O(1)

Output:

```
PROBLEMS   OUTPUT   TERMINAL   JUPYTER   DEBUG CONSOLE

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\trupti patil\OneDrive\Desktop\ACADEMICS\SEM5\DAA\ExpQ>
f ($?) { g++ A6Q2c.cpp -o A6Q2c } ; if ($?) { .\A6Q2c }
The maximum profit is 31.5
PS C:\Users\trupti patil\OneDrive\Desktop\ACADEMICS\SEM5\DAA\ExpQ>
```