SEPTEMBER 15,2020

DESIGN AND ANALYSIS OF ALGORITHMS

(QUESTION: 5.1, ACTIVITY SELECTION)
EXERCISE 5

SUBMITTED BY-
NAME - RAJ KRISHNA
ROLL NO - 1805233
BATCH - CSE-G1
GROUP - B

1. The objective of the Experiment

The objective of the experiment is to select the **non-conflicting activities** that need to be executed by a single person or machine in a given time frame by **greedy technique**.

2. Solution Code

```
#include<bits/stdc++.h>
using namespace std;
bool comp(pair<int,int>i,pair<int,int>j){
	return i.second<j.second;
}
int main()
{
	vector<pair<int,int>>vec(11);
	cout<<"Starting time of activities: "<<endl;
	for(int i=0;i<11;i++){
		cin>>vec[i].first;
	}
	cout<<"Finishing time of activities: "<<endl;
	for(int i=0;i<11;i++){
		cin>>vec[i].second;
	}
	sort(vec.begin(),vec.end(),comp);
	vector<pair<int,int>>v;
	v.push_back(vec[0]);
	pair<int,int>current=vec[0];
	for(int j=1;j<11;j++){
		if(vec[j].first > current.second){
			v.push_back(vec[j]);
			current=vec[j];
			}
	}
```

```
        vector<pair<int,int>>::iterator i;
        cout<<"Following activities are selected- "<<endl;
        for(i=v.begin();i!=v.end();i++){
                cout<<"("<<(*i).first<<","<<(*i).second<<"), ";
        }
        cout<<endl;
  return 0;
}
```

3. Summary of the program

The **activity selection** problem is an **optimization problem** concerning the selection of **non-conflicting activities** to perform within a given time frame, given a set of activities each marked by a **start time** and **finish time**. We have to select the maximum number of activities that can be performed by a single person or machine, assuming that a person can only work on a single activity at a time.

Assume there exist **n activities** with each of them being represented by a **start time** $\mathbf{s_i}$ and **finish time** $\mathbf{f_i}$. Two activities i and j are said to be **non-conflicting** if $\mathbf{s_i \geq f_j}$ or $\mathbf{s_j \geq f_i}$.
The activity selection problem is solved using a **greedy algorithm** to find a solution will always result in an **optimal solution**.

Now, the greedy approach for this problem,

- ➔ First, we need to sort the activities in ascending order according to their finishing time.
- ➔ Then, select the first activity from the sorted array and print it.

Then, do the following for remaining activities in the sorted array.

➔ Check, if the starting time of this activity is greater than the finishing time of previously selected activity then select this activity and print it.

Best Case - $O(n*\log_2 n)$
Average Case - $O(n*\log_2 n)$
Worst Case - $O(n*\log_2 n)$

4. Sample Output

Starting time of activities:
0 1 2 3 3 5 5 6 8 8 12

Finishing time of activities:
6 4 14 5 9 7 9 10 11 12 16

Following activities are selected-
(1,4), (5,7), (8,11), (12,16),