Job Scheduling using Greedy Approach:

In job scheduling, tasks with earliest deadlines are prioritized. The greedy approach selects the job with the earliest deadline first.

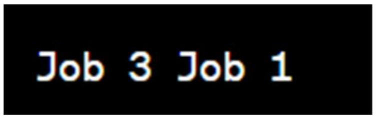
Algorithm:

1. Sort jobs based on deadlines.
2. Initialize the result array and a Boolean array to track job occupation.
3. Iterate through sorted jobs. If a slot is found, assign the job to it and mark the slot as occupied.

Code:

```
#include <iostream>
#include <algorithm>
using namespace std;

struct Job {
   int id, deadline, profit;
};

bool comparison(Job a, Job b) {
   return (a.profit > b.profit);
}

void jobScheduling(Job arr[], int n) {
   sort(arr, arr + n, comparison);
   int result[n];
   bool slot[n];
   fill(slot, slot + n, false);

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

   for (int i = 0; i < n; i++) {
      if (slot[i])
         cout << "Job " << arr[result[i]].id << " ";
   }
}

int main() {
   Job arr[] = {{1, 4, 20}, {2, 1, 10}, {3, 1, 40}, {4, 1, 30}};
   int n = sizeof(arr) / sizeof(arr[0]);
   jobScheduling(arr, n);
   return 0;
}
```

Output:



```
Job 3 Job 1
```

Space complexity: ___________

Justification:____
____

Time complexity:

Best case time complexity: ___________

Justification:____
____

Worst case time complexity: ___________

Justification:____