GeeksforGeeks

A computer science portal for geeks

GeeksQuiz

Greedy Algorithms | Set 1 (Activity Selection Problem)

Greedy is an algorithmic paradigm that builds up a solution piece by piece, always choosing the next piece that offers the most obvious and immediate benefit. Greedy algorithms are used for optimization problems. An optimization problem can be solved using Greedy if the problem has the following property: *At every step, we can make a choice that looks best at the moment, and we get the optimal solution of the complete problem*.
If a Greedy Algorithm can solve a problem, then it generally becomes the best method to solve that problem as the Greedy algorithms are in general more efficient than other techniques like Dynamic Programming. But Greedy algorithms cannot always be applied. For example, Fractional Knapsack problem (See this) can be solved using Greedy, but 0-1 Knapsack cannot be solved using Greedy.

Following are some standard algorithms that are Greedy algorithms.
1) Kruskal's Minimum Spanning Tree (MST): In Kruskal's algorithm, we create a MST by picking

edges one by one. The Greedy Choice is to pick the smallest weight edge that doesn't cause a cycle in the MST constructed so far.
2) Prim's Minimum Spanning Tree: In Prim's algorithm also, we create a MST by picking edges one by one. We maintain two sets: set of the vertices already included in MST and the set of the vertices not yet included. The Greedy Choice is to pick the smallest weight edge that connects the two sets.
3) Dijkstra's Shortest Path: The Dijkstra's algorithm is very similar to Prim's algorithm. The shortest path tree is built up, edge by edge. We maintain two sets: set of the vertices already included in the tree and the set of the vertices not yet included. The Greedy Choice is to pick the edge that connects the two sets and is on the smallest weight path from source to the set that contains not yet included vertices.
4) Huffman Coding: Huffman Coding is a loss-less compression technique. It assigns variable length bit codes to different characters. The Greedy Choice is to assign least bit length code to the most frequent character.

The greedy algorithms are sometimes also used to get an approximation for Hard optimization problems. For example, Traveling Salesman Problem is a NP Hard problem. A Greedy choice for this problem is to pick the nearest unvisited city from the current city at every step. This solutions doesn't always produce the best optimal solution, but can be used to get an approximate optimal solution.

Let us consider the Activity Selection problem as our first example of Greedy algorithms. Following is the problem statement.
You are given n activities with their start and finish times. Select the maximum number of activities that can be performed by a single person, assuming that a person can only work on a single activity at a time.
Example:

```
Consider the following 6 activities.
     start[]  =  {1, 3, 0, 5, 8, 5};
     finish[] =  {2, 4, 6, 7, 9, 9};
The maximum set of activities that can be executed
by a single person is {0, 1, 3, 4}
```

The greedy choice is to always pick the next activity whose finish time is least among the remaining activities and the start time is more than or equal to the finish time of previously selected activity. We can sort the activities according to their finishing time so that we always consider the next activity as minimum finishing time activity.

1) Sort the activities according to their finishing time
2) Select the first activity from the sorted array and print it.
3) Do following for remaining activities in the sorted array.
…….a) If the start time of this activity is greater than the finish time of previously selected activity then select this activity and print it.

In the following C implementation, it is assumed that the activities are already sorted according to their finish time.

```
#include<stdio.h>

// Prints a maximum set of activities that can be done by a single
// person, one at a time.
//  n   -->  Total number of activities
//  s[] -->  An array that contains start time of all activities
//  f[] -->  An array that contains finish time of all activities
void printMaxActivities(int s[], int f[], int n)
{
    int i, j;

    printf ("Following activities are selected \n");

    // The first activity always gets selected
    i = 0;
    printf("%d ", i);

    // Consider rest of the activities
    for (j = 1; j < n; j++)
    {
      // If this activity has start time greater than or equal to the finish
      // time of previously selected activity, then select it
      if (s[j] >= f[i])
      {
          printf ("%d ", j);
          i = j;
      }
    }
}

// driver program to test above function
int main()
{
    int s[] =  {1, 3, 0, 5, 8, 5};
    int f[] =  {2, 4, 6, 7, 9, 9};
    int n = sizeof(s)/sizeof(s[0]);
    printMaxActivities(s, f, n);
    getchar();
    return 0;
}
```

Output:

```
Following activities are selected
0 1 3 4
```

How does Greedy Choice work for Activities sorted according to finish time?
Let the give set of activities be S = {1, 2, 3, ..n} and activities be sorted by finish time. The greedy choice is to always pick activity 1. How come the activity 1 always provides one of the optimal solutions. We can prove it by showing that if there is another solution B with first activity other than 1, then there is also a solution A of same size with activity 1 as first activity. Let the first activity selected by B be k, then there always exist A = {B – {k}} U {1}.(Note that the activities in B are independent and k has smallest finishing time among all. Since k is not 1, finish(k) >= finish(1)).

References:

Introduction to Algorithms by Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein
Algorithms by S. Dasgupta, C.H. Papadimitriou, and U.V. Vazirani
http://en.wikipedia.org/wiki/Greedy_algorithm

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

Related Topics:

- Linearity of Expectation
- Iterative Tower of Hanoi
- Count possible ways to construct buildings
- Build Lowest Number by Removing n digits from a given number
- Set Cover Problem | Set 1 (Greedy Approximate Algorithm)
- Find number of days between two given dates
- How to print maximum number of A's using given four keys
- Write an iterative O(Log y) function for pow(x, y)

Tags: Activity Selection Problem, Greedy, Greedy Algorithm

Tweet 2 +1 3

Writing code in comment? Please use **ideone.com** and share the link here.

49 Comments GeeksforGeeks 1 Login

Recommend Share Sort by Newest

Join the discussion…

d_k • 2 months ago

For input

int s[] = {1, 3, 5, 6, 8, 4};

int f[] = {2, 4, 9, 7, 9, 9};

your code give output 0,1,2 means total activity =3
But he can perform 4 activity 0,1,3,4

Reply • Share ›

devakar verma ➦ d_k • a month ago

Greedy algorithm doesn't guarantee optimal solution

Reply • Share ›

Sanket Patel ➦ d_k • 2 months ago

First you have to sort activities by their finish times.

Reply • Share ›

Asim Haroon ➦ Sanket Patel • 2 months ago suppose i don't have a sorted activity list, how can i sort the finish time array and also manage the start time array accordingly

• Reply • Share ›

Sanket Patel Asim Haroon • 2 months ago

Pass in both arrays as arguments to the sorting algorithm.
Use comparisons only between the elements of the finish times array.
Perform swaps in both arrays instead of swaps in only the finish array.

• Reply • Share ›

H@RUN • 4 months ago

in activity selection problem,if we sort it by starting time,why it doesnot give correct answer?

• Reply • Share ›

Ashu H@RUN • 3 months ago

You can sort based on the start time and then pick the activities in the reverse order, given the condition that the finish time of activity next picked is less than the start time of the previously picked activity.
The point here is that start time and end time are two events and you want to choose based on an event that gives you some information about the overlap an activity can have with another activity. If you start looking from the beginning of activities, the start time does not give you any information about how long this activity would extend (what is it's finish time). However, when you choose from the end of activities, start time gives you this information that the finish time of the chose activity is within a certain bound (less than the finish time of the last picked activity).

It's like this : there is a valley with surrounded by hills on the two sides. When you are on the left hill, you would stand on it's right end to look down. However, when you are on the right hill, you would stand on it left end.

• Reply • Share ›

lasan H@RUN • 4 months ago

same doubt mc

• Reply • Share ›

neelabhsingh • 5 months ago

Please check my code http://ideone.com/T1inmn, Here actives are not sorted according to their finish time,
There is one trick is used in GeeksForGeeks, which is following,
In greedy approach we always print first activity, This activity always performed, But This gives confusion that, if first activity finish time is more than two activity which start after first activity.
for Example:
number of activity is 3
1 2 4

6 3 5

So only one activity will be performed, But there are two activity that can performed.
Here we forget that activity are sorted by their finish order.
So
2 4 1
3 5 6

So number of activity is two.

Now if we print first activity inside the for loop then last activity will not print, because condition if (s[j] >= f[i]) will fail, So we don't know which activity,

So print first activity outside for loop, Now in for loop activity which start time print, if (s[j] >= f[i]) print j activity, because i already printed.

2 ⌃ | ⌄ • Reply • Share ›

Anand Barnwal • 5 months ago

Here is the the implementation using qsort function when activities are not sorted.

```
#include <stdio.h>
#include <stdlib.h>

int cmp( const void *a, const void *b )
{
int *A = (int *)a;
int *B = (int *)b;
return A[1] - B[1]; //Here 1 is index of finish
}

int main(void)
{
int start_finish[10][2], num_activity, i, j;
printf("Enter the number of activities: ");
scanf("%d", &num_activity);
printf("Enter the start and finish of all activities: \n");
for(i = 0; i < num_activity; i++)
```

see more

⌃ | ⌄ • Reply • Share ›

guest0129012 • 6 months ago

what is the time complexity of the above algorithm?

1 ⌃ | ⌄ • Reply • Share ›

Goku ➦ guest0129012 • 6 months ago

O(N) where N is the number of activities.

• Reply • Share ›

Abhi Goku • 5 months ago

O(NlogN) if the original array is unsorted

1 • Reply • Share ›

Sumit Jain • 7 months ago

Can you tell me please that instead of selecting the first activity to finish, selecting the last activity to start that is compatible with all previously selected activities, what will be the result. Please provide an example. pls

• Reply • Share ›

sheshu Sumit Jain • 7 months ago

if u have already sorted according to finish time then your selection process will work properly.

but if not sorted then

eg: s = {1,3,5,7,9} f={3,10,7,9,10} will give {1,3} in your selection process but actual answer is {1,5,7,9}

• Reply • Share ›

debanjan sarkar • 7 months ago

what is the time complexity--------------

• Reply • Share ›

BaoNC debanjan sarkar • 7 months ago

I think it is O(n).

• Reply • Share ›

Harish BaoNC • 7 months ago

It would be O(nlgn) as it also involves sorting.

• Reply • Share ›

debanjan sarkar Harish • 7 months ago

thank you

• Reply • Share ›

Sumit Jain Harish • 7 months ago

Can you tell me please that selecting the last activity to start that is compatible with all previously selected activities, what will be the result. Please provide an example. pls

• Reply • Share ›

Abhinav Bhardwaj • 8 months ago

Java Implementation of Activity Scheduling Algorithm.

```
package Greedy;

import java.util.HashMap;
import java.util.Iterator;
import java.util.TreeMap;

public class ActivitySelection {

public void printActivity(int[] startTime, int[] finishTime) {
HashMap<integer, integer=""> myMap = new HashMap<integer, integer="">();
//Putting all the finish,StartTime Pairs in a HashMap.
for (int i = 0; i < startTime.length; i++) {
myMap.put(finishTime[i], startTime[i]);
}

//Sorting the map according to the finish Time and mainlining the relation with the start Time.

TreeMap<integer, integer=""> treeMap = new TreeMap<integer, integer="">(myMap);
```

see more

^ | ⌄ • Reply • Share ›

cache ➦ Abhinav Bhardwaj • 7 months ago

This code won't work when keys ie finish time in this case are not unique

2 ^ | ⌄ • Reply • Share ›

ashish • 9 months ago

you can sort finishing time array keeping track with their respective starting time by this algo. i am printing starting time instead of indices of starting time for more clearification.

```
#include<stdio.h>
void swap(int *a,int *b)
{
int temp=*a;
*a=*b;
*b=temp;
}
void printfmaxact(int a[],int b[],int n)
{
int i,j;
i=0;
printf("%d ",a[i]);
```

```
for(j=1;j<n;j++) {="" if(a[j]="">=b[i])
{
printf("%d ",a[j]);
```

see more

^ | ⌄ • Reply • Share ›

Puneet • 9 months ago

java implementation -:

```
import java.util.ArrayList;

import java.util.Collections;

import java.util.List;

import java.util.Scanner;

public class ActivitySelection {

public static void main(String args[])

{

/* String input = "5+20+11+1";

StringBuilder sb = new StringBuilder();

List<integer> list1 = new ArrayList<integer>();
```

see more

^ | ⌄ • Reply • Share ›

pango89 • 9 months ago

Here is a C++ Implementation using STL

http://ideone.com/Ty1edb

5 ^ | ⌄ • Reply • Share ›

neelabhsingh • a year ago

@GeeksforGeeks In the above solution: We assume that activity is sorted by finish time, there is no procedure in the above how to sort the finish time along with the activity. If we only sort the finish time then we can't track the activity number.

```
int s[] = {1, 3, 0, 2,3 , 5};
int f[] = {9, 5, 6, 7, 6, 7};
```

This can not be solved in above approach.

```
3 0 3 2 5 1
5 6 6 7 7 9 // Sort by finish time along with the start time
```

5 6 6 7 7 9 // Sort by finish time along with the start time.

Ans: 2 activity

• Reply • Share ›

decade ➦ neelabhsingh • 8 months ago

answer is 1 activity.(with starting time 3 and ending time 5). starting time of next activity should be strictly greater than the finish time of previous one.

• Reply • Share ›

guest • a year ago

can someone post links to some problem on this topic??

• Reply • Share ›

12rad ➦ guest • a year ago

http://community.topcoder.com/...

1 • Reply • Share ›

new • a year ago

What is we also need to give the sequence number of the activities we have selected.We preprocess to sort the arrays but the sequence number should be on the basis of the initial array

3 • Reply • Share ›

RAJAT • 2 years ago

WHAT IF THE FINISHING TIME OF THE FIRST JOB WAS NOT 2 ,BUT SAY 8..THEN IN THIS CASE IT WOULD NOT HAVE GIVEN THE CORRECT ANS

IT GIVES 0 4 BUT THE BEST CASE CAN BE THAT OF 1 3 4

• Reply • Share ›

Sree Harsha Konduri ➦ RAJAT • a year ago

The finish list is to be sorted first. So when finish time of the first job is 8 it will be placed between activities 5-7 and 8-9.

11 • Reply • Share ›

arun • 2 years ago

Assuming that a person can work on only a single task at a time how can the following set be a solution

{0, 1, 3, 4}

The start time of 1 is before the end time of 0 which is 6.

7 • Reply • Share ›

Aadi ➦ arun • 2 years ago

here 0 means the 0th job which is the first job above whose starting time is 1.

2 ^ | ⌄ • Reply • Share ›

Dasharath • 2 years ago

if the input is
int s[] = {1, 3, 0, 5, 8, 7, 9};
int f[] = {2, 4, 6, 7, 9, 9, 10};

This algorithm gives (0,1,3,4) as output. The output should rather be (0,1,3,5,6)

2 ^ | ⌄ • Reply • Share ›

kartik ➦ Dasharath • 2 years ago

I think the algorithm will give {0, 1, 3, 4, 6} which is also a correct answer.

7 ^ | ⌄ • Reply • Share ›

soha • 3 years ago

How can greedy algorithms help in complex problems like shortest path problems?

^ | ⌄ • Reply • Share ›

Me • 3 years ago

How can this algorithm be modified such that we need to choose maximum number of tasks?

For eg.
s[] = { 1, 3, 5, 7, 9}
f[] = {10, 4, 6, 8, 10}

As per this algorithm, only task 0 is printed in the output.
But optimally, if we need to do the maximum number of tasks, output would be, 1, 2, 3, 4

How can this algorithm be changed for this kind of modification?

^ | ⌄ • Reply • Share ›

Rakesh ➦ Me • a year ago

3 5 7 9 1

4 6 8 10 10

^ | ⌄ • Reply • Share ›

Sree Harsha Konduri ➦ Me • a year ago

We need to preprocess the inputs before we can use the greedy algorithm of activity selection. The finish array needs to be sorted in increasing order to run this algorithm. Then it is intuitive to pick the activity with the smallest finish time which does not overlap with running activities.
After preprocessing
s[] = { 3 5 7 1 9}
f[] = { 4 6 8 10 10}

1 ⌃ | ⌄ • Reply • Share ›

Guest ➦ Me • 2 years ago

why only task 0 is printed. if you sort the finish times then your input becomes
S[] = {3,5,7,1,9} ,
f[] = {4,6,8,10,10} in which case your op will be 0,1,2,4

1 ⌃ | ⌄ • Reply • Share ›

agmafara ➦ Me • 3 years ago

Enhancement to last suggestion.
Your view?

```
#include
#include

void printMaxActivities(int s[], int f[], int n) {
int i, j;

printf("Following activities are selected \n");

// The first activity always gets selected
i = 0;
printf("%d ", i);

// Consider rest of the activities
for (j = 1; j = f[i]) {
printf("%d ", j);
i = j;
```

see more

1 ⌃ | ⌄ • Reply • Share ›

agmafara ➦ Me • 3 years ago

```
#include
#include

void printMaxActivities(int s[], int f[], int n) {
int i, j;

printf("Following activities are selected \n");

// The first activity always gets selected
i = 0;
printf("%d ", i);

// Consider rest of the activities
for (j = 1; j = f[i]) {
printf("%d ", j);
i = j;
}
}
}
```

see more

⌃ | ⌄ • Reply • Share ›

kartik ➦ Me • 3 years ago

The implementation given in the post assumes that the input tasks are sorted according to their finish time.

For unsorted inputs, you need to add a preprocessing step that first sorts the given tasks ccording to their finish time.

1 ⌃ | ⌄ • Reply • Share ›

Mad Coder • 3 years ago

In your explanation of algorithm for activity selection problem,in 3rd point it is written that

If the start time of this activity is less than the finish time of previously selected activity then select this activity and print it.

It should rather be if start time of current activity is greater than the finish time of previous activity, then current activity should be selected otherwise both will overlap.

1 ⌃ | ⌄ • Reply • Share ›

GeeksforGeeks ➦ Mad Coder • 3 years ago

@Mad Coder: Thanks for pointing this out. We have updated the post. Keep it up!

⌃ | ⌄ • Reply • Share ›

cracker • 3 years ago

Another great article. Keep rocking GeeksforGeeks. Please post an article on Kruskal, how to implement it.

⌃ | ⌄ • Reply • Share ›

PsychoCoder ➦ cracker • 3 years ago

@Cracker :

http://geeksforgeeks.org/forum...

may be this can help you. :)

1 ⌃ | ⌄ • Reply • Share ›

Rute ➦ PsychoCoder • 3 years ago

Any good search manektirg campaign is a multifaceted approach to reach on top of the rankings.a0Many SEO factors such as site structure, content relevance, and submission of press releases online will influence the position of your website / blog in search engine results.a0From 2011, the engagement is a key metric that can be measured by activity ona0social networksa0and how long visitors spend on your site / blog, here are some changes thata0Google Pandaa0brought.

^ | ˅ • Reply • Share ›

✉ **Subscribe** **Add Disqus to your site** **Privacy**

-
-
-
- - Interview Experiences
  - Advanced Data Structures
  - Dynamic Programming
  - Greedy Algorithms
  - Backtracking
  - Pattern Searching
  - Divide & Conquer
  - Mathematical Algorithms
  - Recursion
  - Geometric Algorithms
-

Popular Posts

- All permutations of a given string
- Memory Layout of C Programs
- Understanding “extern” keyword in C
- Median of two sorted arrays
- Tree traversal without recursion and without stack!
- Structure Member Alignment, Padding and Data Packing
- Intersection point of two Linked Lists
- Lowest Common Ancestor in a BST.
- Check if a binary tree is BST or not
- Sorted Linked List to Balanced BST

Follow @GeeksforGeeks

Recent Comments

- lt_k

  i need help for coding this function in java...

  Java Programming Language · 1 hour ago

- Piyush

  What is the purpose of else if (recStack[*i])...

  Detect Cycle in a Directed Graph · 1 hour ago

- Andy Toh

  My compile-time solution, which agrees with the...

  Dynamic Programming | Set 16 (Floyd Warshall Algorithm) · 1 hour ago

- lucy

  because we first fill zero in first col and...

  Dynamic Programming | Set 29 (Longest Common Substring) · 2 hours ago

- lucy

  @GeeksforGeeks i don't n know what is this long...

  Dynamic Programming | Set 28 (Minimum insertions to form a palindrome) · 2 hours ago

- manish

  Because TAN is not a subsequence of RANT. ANT...

  Given two strings, find if first string is a subsequence of second · 2 hours ago

-