Design and Analysis of Algorithms

5.2 Greedy Algorithms

Daniel Shannon

May 3rd, 2022

5.2.2

Suppose we find a minimum spanning tree T in a graph.

Suppose we then add 1 to the weight of every edge in the graph.

Is T still a minimum spanning tree? Why or why not?

Yes T is still a minimum spanning tree because adding 1 to the weight will not

1. create cycles
2. change the ordering of the edge weights

5.2.4

Assign the following strings:

- A: 0
- B: 00
- C: 01
- D: 1

Then AABDC = 0000101

What is the problem with this?

The problem is that C could be mistaken for AD, or B could be AA, or BC could be AAAD etc.

5.2.6

- You are given a collection of items
- Each item i has a weight w_i and value v_i
- You have a bag that can hold a total weight of W
- You want to maximize the value of the items in your bag
- Design an algorithm to decide which items to pick

My Solution

```
1  Procedure Knapsack(f)
2  Input: array f[1...n] of items (v_i,w_i)
3  Output: An encoding tree with n leaves
4
5  Let H be a priority queue of v_i:w_i ratios (r_i), ordered by
       r_i
6  for i = 1 to n: insert(H,i)
7  for k = k+1 to 2n-1:
8    i=deletemin(H) j=deletemin(H)
9    create a node numbered k with children i,j
10   f[k]=f[i]/f[j]
11   insert(H,k)
```

Solution

1. sort all items by r_i
2. add items in order until space is gone
3. at the end if can't add a full item add as much as you can
4.

5.2.8

> Same as before, except you cannot take fractional items. Does your algorithm always find the optimal solution? Explain why, or give a counterexample.

No this does not find the optimal solution. Rather than taking a portion of the next smallest r_i, we can take several less valuable items with less weight. So we could skip items until we find ones that fit!

5.2.10

> - Given a set of elements B, and sets $S_1, S_2...S_k$ that are subsets of B
> - How many sets S_i, do you need to pick so that every element from B appears in at least one selected set?
>
> Can you develop a greedy algorithm to solve the set cover problem?

1. select a starting town
2. determine which towns are out of range, and mark as unexplored
3. place a school in the current town
4. recursively explore the out of range towns