

# Greedy algorithms and matroids

Éric Schost

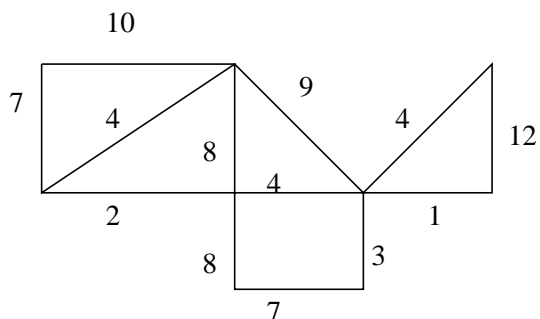
September 27, 2007

## 1 What to read

My main reference was [1]. Greedy algorithms and matroids are nicely described in Chapter 16 of [1]; there is a much more mathematically-oriented presentation in Chapter 1 of [2]. Spanning trees are discussed in depth in Chapter 23 of [1]; union-find (and other algorithms for disjoint sets) are in Chapter 21 of [1].

## 2 Problems

1. Find a maximum spanning tree for the following graph using the greedy algorithm. Represent the set of edges under construction at each step of the algorithm.



2. Give the cost of the greedy algorithm for the second car rental problem (where we want to maximize the number of rentals), assuming that all operations (comparing integers, copying integers, ...) have cost 1. I'm only asking for estimates like  $O(\dots)$ .
3. Write a *clean* proof that the greedy algorithm is optimal for the second form of the car rental problem. Since this can be tricky to write in general, I'm only asking you to do it (as I did) for  $q = 4$ , using the notation in the slides (that is, prove that in this case,  $p \geq 4$ ).
4. Give an algorithm for computing minimum spanning trees and prove its correctness. Hint: try to re-use the algorithm for maximum spanning trees.

5. Many greedy algorithms require to sort (once and for all) the set from which we pick our elements (edges in a graph, requests for the car rental problem, ...). For sorting  $n$  elements, general algorithms have at best a complexity of  $O(n \log(n))$ . Supposing that the elements to sort all have integer weight in  $1, \dots, \ell$ , you are asked to find a sorting algorithm of complexity  $O(n + \ell)$ , and to justify its running time.

The input is the array  $w[1], \dots, w[n]$  of the weights. The output is an array containing the same values, sorted in increasing order.

Hint: count elements of weight  $1, 2, \dots, \ell$ .

6. We consider the problem of making change: given an amount  $S$ , we want to make change using the fewest number of coins. The types of coins we can use are given (for instance: 1 dollar, 25 cents, 1 cent); we have an unlimited amount of coins of each type. For simplicity, we express all amounts in cents.

Suppose we have  $n$  types of coins; the values are stored in an array  $v$ , by decreasing order:  $v[1] > v[2] \dots > v[n]$ . Given the amount  $S$  (which is an integer number), the greedy algorithm is the following:

- Set  $g[1] = \dots = g[n] = 0$  ( $g[i]$  tells us how many coins of value  $v[i]$  we use).
- for  $i = 1, \dots, n$ , do
  - while  $S \geq v[i]$ , let  $S = S - v[i]$  and  $g[i] = g[i] + 1$ .

Give the steps of the algorithm for  $n = 3$ ,  $v[1] = 10$ ,  $v[2] = 5$ ,  $v[3] = 1$  and  $S = 38$ .

Prove that if  $v[n] = 1$ , the greedy algorithm always gives you the exact change.

Give an example, with  $v[n] < 1$ , where the greedy algorithm fails to give the exact change, even though a solution with the exact change exists. You don't have to use Canadian coins, any coins will do.

Prove that the greedy algorithm is optimal (gives the smallest possible number of coins) for  $n = 2$ ,  $v[1] = 2$  and  $v[2] = 1$ . *Difficult extra question: do the same for  $n = 3$ ,  $v[1] = 4$ ,  $v[2] = 2$  and  $v[3] = 1$ . You do not have to do this question to get full marks on this assignment.*

Give an example, with  $v[n] = 1$ , where the greedy algorithm does not give the optimal answer.

## References

- [1] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms (Second Edition)*. MIT Press, 2001.
- [2] J. Lee. *A first course in combinatorial optimization*. Cambridge University Press, 2004.