

## Greediness

Reading: *CLR* 17.1 – 17.3

---

### Hill Climbing

You are trying to climb to the top of a hill in a dense fog. A **greedy** strategy is to take each step in the direction of the highest gradient.

Whether or not the greedy strategy works depends on characteristics of the hill. Explain.

---

### Coin Changing

You want to make change for a given amount using a minimal number of coins that range over a given list of denominations (e.g., [25,10,5,1]). A greedy strategy is to choose for the next coin the largest denomination that is less than the remaining amount.

```
Greedy-Make-Change(amount, denoms)
  coins ← {} ▷ This is a bag, not a set.
  while amount > 0 do
    if Empty?(denoms) then
      error "Greedy strategy fails"
    let h = Head(denoms) in
      if h ≤ amount then
        coins ← coins ∪ {h}
        amount ← amount - Head(denoms)
      else
        denoms ← Tail(denoms)
  return coins
```

The greedy strategy works for some denomination lists (e.g., [25,10,5,1]) but not others (e.g., [25,10,1]).

---

### Generic Greedy Algorithm

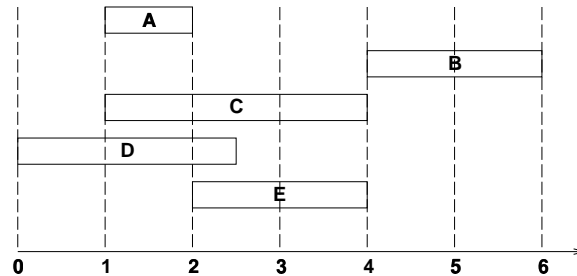
A greedy algorithm makes the locally optimal choice at every step:

```
Greedy(problem)
  soln ← {}
  subproblem ← problem
  while not Solution?(soln, problem) do
    choice ← Greedy-Choice(subproblem)
    ▷ Greedy-Choice makes locally optimal choice for current subproblem.
    soln ← choice ∪ soln
    subproblem ← Simplify(subproblem, choice)
    ▷ Simplify gives remaining subproblem after choice is made.
  return soln
```

---

## Activity Selection Problem

Given a set of activities for a given room with start and stop times, schedule maximal number of activities. E.g.



Greedy strategy: schedule by sorted stop times.

---

## Proving Greedy Strategy Optimal

One approach is to show algorithm has the following two properties:

1. **Greedy Choice Property:** An optimal solution can begin with the greedy choice. (Note: there may be many optimal solutions.)
2. **Optimal Substructure Property:** The optimal solution for the whole problem can be derived from the optimal solution for the parts.

This is usually shown as follows. Assume that you are given an optimal solution  $S$  for the whole problem and show that you can extract the optimal solution  $S_i$  for subproblem  $i$  from the optimal solution to the whole problem. The proof proceeds by contradiction: assume that there is a better solution  $S'_i$  to subproblem  $i$  and show that you could construct a better whole solution  $S'$  using  $S'_i$ . But that contradicts your original assumption about the optimality of  $S$ . So it must be the case that your other assumption is incorrect:  $S'_i$  cannot be better than  $S_i$ . That is, each  $S_i$  is optimal.

---

## Knapsack Problems

Given items with integer values and weights, fill a knapsack with weight capacity  $W$  so that it carries the most valuable load.

1. *Fractional knapsack problem:* can take fraction of an item. Greedy strategy is optimal.
2. *0/1 knapsack problem:* must take entire item. Greedy strategy is not always optimal.