Greedy Algorithms

Arnav Gupta

April 9, 2024

Contents

1 Overview

For trying to solve a combinatorial optimization problem:

- have a large, but finite domain $\mathcal{D}$

- want to find an element E in $\mathcal{D}$ that minimizes/maximizes a cost function

1.1 Strategy

- build E step by step
- don't think ahead, just try to improve as much as possible at every step
- simple algorithms, but usually no guarantee to get the optimal
- hard to prove correctness, easy to prove incorrectness

2 Interval Scheduling

2.1 Problem

- input: n intervals $[s_1, f_1], [s_2, f_2], \dots, [s_n, f_n]$
- output: a maximal subset of disjoint intervals

2.2 Algorithm

1. Let S be the empty set
2. Sort the intervals such that $f_1 \leq f_2 \leq \cdots \leq f_n$
3. For i from 1 to n do
   (a) if interval i, $[s_i, f_i]$ has no conflicts with intervals in S
      i. add i to S
4. Return S

2.3 Correctness

2.3.1 The Greedy Algorithm Stays Ahead

Assume O is an optimal solution, the goal is to show $|S| = |O|$.

- Suppose $i_1, i_2, \dots, i_k$ are the intervals in S in the order they were added to S by the greedy algorithm
- Similarly, the intervals in O are denoted by $j_1, \dots, j_m$

- assume that the intervals in O are ordered in the order of the start and finish times
- Prove that $k = m$

2.3.2 Lemma

First consider the lemma: For all indices $r \leq k$ we have $f(i_r) \leq f(j_r)$.

By induction:

- for $r = 1$, the statement is true
- suppose $r > 1$ and the statement is true for $r - 1$
  - we show that the statement is true for r
- by induction hypothesis, $f(i_{r-1}) \leq f(j_{r-1})$
- by the order on O, $f(j_{r-1}) < s(j_r)$
- hence $f(i_{r-1}) < s(j_r)$
- thus, at the time the greedy algorithm chose i_r, the interval j_r was a possible choice
- the greedy algorithm chooses an interval with the smallest finish time
  - so $f(i_r) \leq f(j_r)$

2.3.3 Proof

Theorem: The greedy algorithm returns an optimal solution

Prove by contradiction:

- if the output S is not optimal, then $|S| < |O|$
- i_k is the last interval in S and O must have an interval j_{k+1}
- apply the previous lemma with $r = k$ and we get $f(i_k) \leq f(j_k)$
- we have $f(i_k) \leq f(j_k) < s(j_{k+1})$
- so j_{k+1} was a possible choice to add to S by the greedy algorithm
  - this is a contradiction by how the greedy algorithm works

3 Interval Colouring

3.1 Problem

- input: n intervals $[s_1, f_1], [s_2, f_2], \ldots, [s_n, f_n]$
- output: use the minimum number of colours to colour the intervals so that each interval gets one colour and two overlapping intervals get two different colours

3.2 Algorithm

1. Sort the intervals by starting time: $s_1 \leq s_2 \leq \cdots \leq s_n$
2. For i from 1 to n do
   (a) use the minimum available colour c_i to colour the interval i (one that doesn't conflict with the colours of intervals already coloured)

3.3 Correctness

Assume the greedy algorithm uses k colours. To prove correctness, the goal is to show that there are no other ways to solve the problem using at most $k-1$ colours.

3.3.1 Proof

Suppose interval ℓ is the first interval to use the colour k:

- interval ℓ overlaps with intervals with colours $1, \ldots, k-1$
- call these intervals $[s_{i_1}, f_{i_1}], [s_{i_2}, f_{i_2}], \ldots, [s_{i_{k-1}}, f_{i_{k-1}}]$
- for $1 \leq j \leq k-1$, $s_{i_j} \leq s_\ell$
- all the intervals overlap with $[s_\ell, f_\ell]$
- since all these intervals overlap with $[s_\ell, f_\ell]$, $s_\ell \leq f_{i_j}$ for $1 \leq j \leq k-1$
- hence s_ℓ is a time contained in k intervals
- so, there is no $k-1$ colouring

4 Minimizing Total Completion Time

4.1 Problem

- input: n jobs, each requiring processing time p_i
- output: an ordering of the jobs such that the total completion time is minimized

4.2 Algorithm

- order the jobs in non-decreasing processing times

4.3 Correctness

- let $L = [e_1, \ldots, e_n]$ be an optimal solution (as a permutation of $[1, \ldots, n]$)
- suppose that L is not in non-decreasing order of processing times
  - so there exists i such that $t(e_i) > t(e_{i+1})$
- sum of the completion times of L is $nt(e_1) + (n-1)t(e_2) + \cdots + t(e_n)$
- contribution of e_i and e_{i+1} is $(n-i+1)t(e_i) + (n-i)t(e_{i+1})$
- let L' be the permutation with e_i and e_{i+1} switched
- their contribution becomes $(n-i+1)t(e_{i+1}) + (n-i)t(e_i)$
- nothing else changes so $T(L') - T(L) = t(e_{i+1}) - t(e_i) < 0$ which is a **contradiction**