Tutorial 2

Gidon Rosalki

2024-11-14

1 Optimisation problem

Definition 1.1. *A problem for calculating the optimal solution between a selection of correct solutions.*

In order to show that our solution is optimal, we need the following:

1. Correctness - our solution is correct
2. Optimality - Our solution is no less good then every other solution

2 Greedy algorithms

Definition 2.1 (Not formal)**.** *An iterative algorithm will be called greedy if every iteration it chooses the best option at that that specific moment*

2.1 Small fuel tank problem

Definition 2.2. *We have a car which travels along a straight line from a_1 to a_n. Across the path there are petrol stations, and the target is to make it to the destination, with a minimal number of stops at petrol stations.*

Input: $a_1, \ldots, a_n \in \mathbb{N}$ - location of petrol stations. $N \in \mathbb{N}$, the number of kilometres that it is possible to travel on a full tank.
Output: A subseries of petrol stations $B = (b_1, \ldots, b_m)$, such that $a_1 = b_1 \wedge a_n = b_m$, and $\forall 1 \leq i \leq m\ b_i - b_{i-1} \leq N$. We are looking for the minimal B.
Example: $(a_1, \ldots, a_7) = (0, 1, 7, 9, 16, 17, 20)\,,\ N = 10$. A solution would be $B = (a_1, a_4, a_6, a_7)$.

2.1.1 Greedy algorithm solution

- We begin with an empty list $B = \emptyset$
- We will insert a_1 into B
- We will pass over all the stops according to their order for $i = 1, \ldots, n$, and we will add a_i to be if either $a_i = a_n$, or if there is insufficient fuel to make it to the petrol station a_{i+1}. Which is to say, if a_j is the last stop in B, then we will add a_i if $a_{i+1} - a_j > N$.
- We will return B

2.1.2 Proof of correctness

Theorem 1 (Correctness)**.** *The algorithm returns a correct solution.*

Proof. The algorithm adds to the solution a_1, a_n, and therefore the first requirement is met. According to the definition of the algorithm, we stop at a petrol station only if we have insufficient fuel to arrive at the next station. From this, we can extrapolate that we stop at the petrol stations only if we have enough fuel to get to it from the last station at which we stopped. □

Theorem 2 (Optimality)**.** *Proof.* We will write in $B = (b_1, \ldots, b_m)$ the greedy solution. In order to show optimality, we will prove the following theorem:

Theorem 3. *For every $1 \leq k \leq m$ there is an optimal solution similar to $C = (b_1, \ldots, b_k, c_{k+1}, \ldots, c_{n'})$*

Proof Induction on k. Basis: For $k = 1$, from the correctness of the algorithm, $b_1 = a_1$. According to the definition, every correct solution begins with a_1, and is therefore also optimal.
Inductive step: We will assume that the theorem is correct for $k-1$, and that there exists a solution $C = (b_1, \ldots, b_{k-1}, c_k, \ldots, c_n)$. In order to show that the theorem is correct for k, we will split into cases:

- $k = m$: Then from the correctness of C, it is true that $c_{m'} = b_m = a_n$, and from the optimality of C, $m' = k$ since otherwise B is a better solution, in contradiction of the fact that C is optimal.
- $k < m$: We will build the solution $C' = (b_1, \ldots, b_k, c_{k+1}, \ldots, c_{m'})$, and we will prove that this is correct and optimal. From the correctness of B, we will get that for all $1 < i \leq k$ we get $b_i - b_{i-1} < N$. From the correctness of C, we get that for all $k + 1 < i \leq m'$ we get $c_i - c_{i-1} < N$. From this we can show that $c_{k+1} - b_k \leq N$

From the execution of the algorithm, we know that b_k is the furthest stop that we can reach from b_{k-1}. Since C is a correct slution, then $c_k - b_{k-1} \leq N$, and it is possible to reach c_k from b_{k-1}, and from this it is true that $c_k \leq b_k$. In total, we get that $c_{k+1} - b_k \leq c_{k+1} - c_k \leq N$, and is therefore correct. We will aso note that $|C| = |C'|$, and therefore C' is optimal. □

From the theorem, B is optimal. To prove this, from the theorem it is true that for $k = m$, there exists an optimal solution C which passes through the station $b_1, \ldots, b_m$. Since B is a correct solution, then from the optimality of the solution, it is necessary that $|C| \leq |B|$, we will therefore get in total that $C = B$ □

Above we assumed that there is an optimal solution. For a finite solution list there is always an optimal solution, but for an infinite solution list, this must be proven.

3 Return to the MST

Input: A connected, undirected graph $G = (V, E)$, and the non trivial weight function $w : E \to \mathbb{R}$.
Output: A minimum spanning tree of G.

Last week we spoke of Kruskal's algorithm, a greedy solution:
Let us write $|V| - n \wedge |E| = m$

- We begin $T = \emptyset$
- We will sort the edges in ascending order by weight
- For $i = 1, \ldots m$: If e_i does not create a cycle in T, we will then add it to T
- Return T

Runtime was spoken of in the previous tutorial.

Theorem 4 (Correctness)**.** *The algorithm returns a minimum spanning tree*

Proof . Homework. □

Theorem 5 (Optimality)**.** *Let there be T, and MST returned by the algorithm, with the edges $\{t_1, \ldots, t_{n-1}\}$, sorted by weight, which is to say $\forall i \in [n-1]\ t_i \geq t_{i-1}$. Let us write that T_k is the graph that contains exactly the edges $\{t_1, \ldots, t_k\}$. $\forall 0 \leq k \leq n-1$ there exists an optimal solution S such that $T_k \subseteq S$*

Proof . By induction on k.
Basis: For $k = 0$ we get that $T_0 = \emptyset$, and all the sets of edges that contain the empty set, and this is also correct for every optimal solution.
Inductive step: Let us assume that there exists and MST S, such that for $k - 1$ there exists $T_{k-1} \subseteq S$. We will split into cases:

- $t_k \in S$: Then $T_k \subseteq S$
- $t_k \notin S$: We will look at $S \sqcup \{t_k\}$. The above graph is created by adding an edge to an MST, and therefore it has a cycle within. Since T is a tree, it is impossible that the cycle is contained within the edges of T. Let $e \in S \setminus T$ be an edge in the cycle.
  - $w(e) < w(t_k)$: We will look at the step of the algorithm where we added t_k to the solution. In T at this step, there are only located the edges $t_1, \ldots, t_{k-1}$. Since $w(e) < w(t_k)$, the algorithm passed over the iteration on the edge e before it passed over t_k, but chose not to add it to the solution. From this it is true that e creates a cycle with t_{k-1}, and therefore in S there is a cycle, which is a contradiction to the correctness of S.

– $w(t_k) \leq w(e)$: In this situation we will look at the graph $S' = S \sqcup \{t_k\} \setminus \{e\}$. We will note that S' is a spanning tree since in $S \sqcup \{t_k\}$ is a connected graph, with a single cycle within, and through the removal of the edge e, there are no longer cycles within. We will note that $T_k \subseteq S'$. In total, $w(S') = w(S) + w(t_k) - w(e) \leq w(S)$ From this it is true that $w(S') \leq w(S)$, and therefore S' is optimal.

□

4 The change problem

Input: $c_1 < \cdots < c_n \in \mathbb{N}$ is a set of coins such that $c_1 = 1$. $k \in \mathbb{N}$, the change that needs to be given back in coins.
Output: The change k elements through a minimum number of coins.

Example: $c = (1, 2, 5, 10)$, $k = 13 \implies 10 + 2 + 1 \wedge k = 14 \implies 10 + 2 + 2$.

Suggested algorithm: At every iteration, take the largest coin that is still smaller than the total change left to be returned, and take of it as much as you can. This will work for some sets, eg $C = (1, 2, 5, 10)$, but not for others for example $C = (1, 15, 25)$. We can see this to be true for $k = 30$, since the algorithm will return $(25 + 1 + 1 + 1 + 1 + 1)$, but the optimal solution is $15 + 15$.