CS 577

# Assignment 4 – More Greedy

Spring 2022

Answer the questions in the boxes provided on the question sheets. If you run out of room for an answer, add a page to the end of the document.

Name: R Ramapriya Wisc id: rrangarath

## More Greedy Algorithms

1. *Kleinberg, Jon. Algorithm Design (p. 189, q. 3).*

   You are consulting for a trucking company that does a large amount of business shipping packages between New York and Boston. The volume is high enough that they have to send a number of trucks each day between the two locations. Trucks have a fixed limit $W$ on the maximum amount of weight they are allowed to carry. Boxes arrive at the New York station one by one, and each package $i$ has a weight $w_i$. The trucking station is quite small, so at most one truck can be at the station at any time. Company policy requires that boxes are shipped in the order they arrive; otherwise, a customer might get upset upon seeing a box that arrived after his make it to Boston faster. At the moment, the company is using a simple greedy algorithm for packing: they pack boxes in the order they arrive, and whenever the next box does not fit, they send the truck on its way.

   Prove that, for a given set of boxes with specified weights, the greedy algorithm currently in use actually minimizes the number of trucks that are needed. Hint: Use the stay ahead method.

**Solution:** Claim:- For the same number of trucks greedy algorithm will ship as many boxes as other optimal algorithm, so can be proved by induction

1) consider we only have one truck, then the greedy algorithm is the same as other methods due to the first ship policy.

2) Consider the conclusion holds for k trucks, consider now we have K+1 trucks, the greedy algorithm will pack boxes in order in the (K+1)-th, truck, the item in (K+1)-th is as many as that in optimal algorithm then we prove lemma.

Now consider we have an greedy algorithm require k trucks and an optimal algorithm requires m trucks for the same amount of boxes. Assume $k > m$, then the greedy algorithm will pack all the boxes in 1.. m trucks, which contradicts

Then $k \leq m$ we have greedy algorithm is the optimal solution

2. *Kleinberg, Jon. Algorithm Design (p. 192, q. 8).* Suppose you are given a connected graph $G$ with edge costs that are all distinct. Prove that $G$ has a unique minimum spanning tree.

**Solution:**

lemma :- Let C be any cycle in G, and let e be the most expensive edge of G, then e is not in any MST of G.

We use lemma ↑ and we prove by contradiction.

Consider 2 MST, T and T', $\exists\, e \in T,\ e \notin T'$

·) Denote all the nodes as V we consider S and V\S as 2 connected components after removing e from T

·) Since $e \notin T'$ we know there must exist $e' \in T'$ such that e' connects S and V\S in T'

·) Then we have a cycle containing all the edge from T and T', since the edge are all distinct we have the most expensive edge in this cycle is not in any MST.

·) This contradicts with T and T' are all MST.

We prove G has a unique ~~mon~~ minimum spanning tree.

3. *Kleinberg, Jon. Algorithm Design (p. 193, q. 10).* Let $G = (V, E)$ be an (undirected) graph with costs $c_e \geq 0$ on the edges $e \in E$. Assume you are given a minimum-cost spanning tree $T$ in $G$. Now assume that a new edge is added to $G$, connecting two nodes $v, w \in V$ with cost $c$.

   (a) Give an efficient ($O(|E|)$) algorithm to test if $T$ remains the minimum-cost spanning tree with the new edge added to $G$ (but not to the tree $T$). Please note any assumptions you make about what data structure is used to represent the tree $T$ and the graph $G$, and prove that its runtime is $O(|E|)$.

**Solution:**

Consider the new added edge e', we have v', w' ∈ V at the ends of e'

Initiative :- S = {u'} and array 1, array 2

Consider MST T, : T while v' ∉ S :

explore adjacent node V with DFS

add V to S, add parent of V to array 1 add cost of edge connected to V to array 2.

end.

then we have the path from u' to v', add e' to the T.

We have a cycle C containing e' on T. check whether e' is the most expensive edge in C. If it is, T remains a MST the runtime is O(|V|)

   (b) Suppose $T$ is no longer the minimum-cost spanning tree. Give a linear-time algorithm (time $O(|E|)$) to update the tree $T$ to the new minimum-cost spanning tree. Prove that its runtime is $O(|E|)$.

**Solution:**

From the previous algorithm, we have a cycle containing e' on T. if e' is not the most expensive edge on cycle, we remove the most expensive edge on cycle, we got T' is the MST. The run time is O(|V|)

4. In class, we saw that an optimal greedy strategy for the paging problem was to reject the page the furthest in the future (FF). The paging problem is a classic online problem, meaning that algorithms do not have access to future requests. Consider the following online eviction strategies for the paging problem, and provide counter-examples that show that they are not optimal offline strategies.[1]

(a) FWF is a strategy that, on a page fault, if the cache is full, it evicts all the pages.

**Solution:**

Consider cache $\square\square$ $U = \{a, b, c, d\}$

request : a b c b a

FWF 5 faults

FF 4 faults

(b) LRU is a strategy that, if the cache is full, evicts the least recently used page when there is a page fault.

**Solution:**

Consider cache $\square\square$ $U = \{a, b, c, d\}$

request = a b c a b

LRU 5 faults

FF 4 faults

[1] An interesting note is that both of these strategies are $k$-competitive, meaning that they are equivalent under the standard theoretical measure of online algorithms. However, FWF really makes no sense in practice, whereas LRU is used in practice.