CSCC73 A2

Saad Makrod

September 2022

Q1 Greedy Algorithm to Maximize Campers in Canoes

Part A:

Let $C = 10$ and $W = \{1, 2, 7, 8\}$ where each $w \in W$ is the weight of a camper. Thus in this example there are 4 campers. Furthermore, let S be the set chosen by the algorithm. By the algorithm first the pair chosen will be the two lightest campers. In this case the pair has the camper with weight 1 and the camper with weight 2. Clearly, $1 + 2 < 10$ thus the pair is added to S, meaning W is now $\{7, 8\}$. Now since $7 + 8 = 15$ no other pair can be added to S. Thus the algorithm gives S consisting of one pair. Therefore $|S| = 1$.

However this is not the optimal set. Let the optimal set be denoted by S^*. Clearly it is possible of $|S^*| = 2$ since instead of pairing the two lightest campers we can instead pair a light camper with a heavier camper. So $S^* = \{\{$*camper with weight 1, camper with weight 8*$\}, \{$*camper with weight 2, camper with weight 7*$\}\}$ or $S^* = \{\{$*camper with weight 2, camper with weight 8*$\}, \{$*camper with weight 1, camper with weight 7*$\}\}$. Note that $|S^*| > |S|$ so S cannot be optimal.

Therefore the algorithm cannot always generate an optimal set.

Part B:

This proof will be done using the promising set approach.

Let S be the set proposed by the algorithm, and let S^* be any optimal set. Before proving the algorithm I will prove the following claim:

If the combined weight of a lightest and heaviest camper does not exceed C, there is an optimal set that contains a pair consisting of these two campers

Claim Proof

The proof will be done by contradiction.

Suppose for contradiction that there does not exist any S^* such that the heaviest and the lightest camper are a pair and the combined weight of a lightest and heaviest camper does not exceed C. If the lightest and heaviest camper are not a pair in S^* then there are three cases: neither the lightest and heaviest camper are in S^*, only one of the lightest and heaviest camper is in S^*, or both the lightest and heaviest camper but they are not a pair.

The case where neither the lightest and heaviest camper are in S^* is impossible. This is because it is given that the combined weight of the lightest and heaviest camper does not exceed C, so if they are not already in a pair then that means that another pair can be added to S^*. This implies that S^* is not optimal. Thus there is a contradiction since S^* must be optimal.

In the case where only one of the lightest and heaviest camper is in S^*, I will define the following: let S_i^* be the pair that contains either the lightest or heaviest camper, let x be the camper in S_i^* that is not the lightest or heaviest camper, let y be the lightest or heaviest camper in S_i^*, and let z be the lightest or heaviest camper not in S^*. I know that the weight of y and z does not exceed C so I can swap x with z. Thus I have $S' = (S^* - S_i^*) \cup \{y, z\}$. S' is valid since no pair in S' will exceed C, furthermore S' is optimal since S' was constructed by deleting and adding a pair from S^*. Since I did not change the size of the set I know that S' is optimal. Thus I have constructed an optimal set that contains a pair consisting of the lightest and heaviest camper. However, this is a contradiction since it was assumed that there is no optimal set that contains a pair consisting of the lightest and heaviest camper.

In the last case where both the lightest and heaviest camper are in S^* but they are not a pair I will define the following: let h be the heaviest camper, let x be the partner of h, let l be the lightest camper, and let y be the partner of l. Since h is by definition the heaviest camper I know that *weight of* $y \leq$ *weight of* h. I also know that *weight of* h + *weight of* $x \leq C$ since $\{x, h\}$ is a pair in S^*. Thus I can do the following:

weight of x + *weight of* $h \leq C$

weight of x + *weight of* $y \leq$ *weight of* x + *weight of* $h \leq C$

Thus $\{x, y\}$ is a pair of campers whose weight does not exceed C. I can construct S' as follows $S' = (S^* - \{\{h, x\}, \{l, y\}\}) \cup \{\{l, h\}, \{x, y\}\}$. I know that S' is valid since the weight of each pair does not exceed C. I also know that S' is optimal since it was constructed by removing and adding two pairs from S^* so $|S'| = |S^*|$. Thus I have constructed an optimal set that contains a pair consisting of the lightest and heaviest camper. However, this is a contradiction since it was assumed that there is no optimal set that contains a pair consisting of the lightest and heaviest camper.

Since all three cases resulted in a contradiction it is indeed true that if the combined weight of a lightest and heaviest camper does not exceed C, there is an optimal set that contains a pair consisting of these two campers.

QED Claim

Now I will prove the following lemma:

For every recursive call i of the algorithm there is an optimal set of pairs S_i^* that contains the set of pairs S_i in variable S at the end of that recursive call

Lemma Proof

The proof will be done by induction on i. The basis $i = 0$ is trivial since at the end of iteration 0 S_0 is empty.

For the induction step, assume that S_i is a subset of some optimal set S_i^*, I want to prove that S_{i+1} is a subset of some optimal set S_{i+1}^*. In the $i + 1$ recursive call, there are three cases, no pair is added to S_{i+1}, or the pair $\{l, h\}$ is added to S_i where either $\{l, h\}$ is already in S_i^* or $\{l, h\}$ is not yet in S_i^*.

In the case where no pair is added to S_i and $\{l, h\}$ is already in S_i^*, just let $S_{i+1}^* = S_i^*$. Thus in these cases there is an optimal set of pairs S_{i+1}^* that contains the set of pairs S_{i+1}.

In the case where $\{l, h\}$ is not already in S_i^*, define the set S' to be all the campers not in S_i. I know that S_i is contained by the optimal set S_i^*. This means that there is some optimal ordering of S' such that $\mid S_i \cup$ *optimal ordering of* $S' \mid = |S_i^*|$. Otherwise S_i^* would not be optimal. Consider S', S' is a set of campers on its own. The algorithm will chose $\{l, h\}$ in S' such that l is the lightest camper in S' and h is the heaviest camper in S' where *weight of* l + *weight of* $h \leq C$. By the Claim I proved that if the combined weight of a lightest and heaviest camper does not exceed C, there is an optimal set that contains a pair consisting of these two campers. Thus by the Claim it follows that there must exist some optimal ordering of S' which I will refer to as S'^* such that $\{l, h\} \in S'^*$. Therefore I know that $\mid S_i \cup S'^* \mid = |S_i^*|$. Thus I know that $S_i \cup S'^*$ is optimal. Furthermore I know that $S_{i+1} \subseteq S_i \cup S'^*$ since S_i is part of the union and the new pair $\{l, h\}$ is included in S'^*. Thus in this case I can define $S_{i+1}^* = S_i \cup S'^*$. This is an optimal set that contains all the pairs of S_{i+1}.

This completes the induction step.

QED Lemma

Proof Algorithm Correctness

By the Lemma, there is an optimal set $S^* \supseteq S_k$. I claim that, $S_k = S^*$. Suppose, for contradiction, that there is some pair $j \in S^* - S_k$. Since S^* is optimal (and hence feasible), j is a valid pair, meaning that the campers in j do not appear anywhere else in S^*. Note that this means j is a pair of campers whose weight does not exceed C and is not in S_k. However, at some point j was considered for inclusion in S and by the algorithm, it would have been added to S, contradicting that $j \notin S_k$. Thus, $S_k = S^*$, and so the greedy algorithm returns an optimal set.

QED Algorithm Correctness

Part C:

This algorithm will not work. Consider the following counter example: Let $C = 17$ and $W = \{1, 1, 1, 1, 5, 7, 7, 10\}$ where each $w \in W$ is the weight of a camper. Furthermore, let S be the set chosen by the algorithm. By the algorithm the first set of 4 chosen will be {*camper with weight 1, camper with weight 1, camper with weight 7, camper with weight 10*}, however since the total weight exceeds 17 the algorithm will choose to discard the camper with weight 10 and continue the process with $W = \{1, 1, 1, 1, 5, 7, 7\}$. Now the algorithm will choose the set {*camper with weight 1, camper with weight 1, camper with weight 7, camper with weight 7*}, since this total weight is less than 17 this set will be added to S. Now the algorithm will have $W = \{1, 1, 5\}$ and since $|W| < 4$ the algorithm will stop. Thus the algorithm will give $S = \{\{$*camper with weight 1, camper with weight 1, camper with weight 7, camper with weight 7*$\}\}$.

However this is not the optimal set. Let the optimal set be denoted by S^*, clearly $S^* = \{\{$*camper with weight 1, camper with weight 1, camper with weight 5, camper with weight 10*$\}, \{$*camper with weight 1, camper with weight 1, camper with weight 7, camper with weight 7*$\}\}$. Note that $|S^*| > |S|$ so S cannot be optimal.

Therefore the algorithm cannot always produce the optimal set.

Q2 Modify Dijkstra for Computer Network Data Transmission

Consider the modified Dijkstra algorithm below

$R = \emptyset$

$d(s) = \infty$

$p(s) = NIL$

For all $v \in V, v \neq s$, $d(v) = 0$, $p(v) = NIL$

while $R \neq V$

  $u =$ node not in R with maximum d-value

  $R = R \cup \{u\}$

  for each node v st $(u, v) \in E$

    $w = min(wt(u, v), d(u))$

    if $w > d(v)$

      $d(v) = w$

      $p(v) = u$

In the algorithm each vertex v has two corresponding values, $d(v)$ and $p(v)$. $p(v)$ represents the predecessor of v, meaning the vertex right before v in the $s \to v$ path. $d(v)$ is the transmission rate of the path, meaning $d(v) = tr(s \to v \ path)$. R is the set of "explored" nodes meaning the nodes that the algorithm has found the optimal path to.

The intuitive idea behind the algorithm is to always choose the path with the maximum transmission rate. This is why the algorithm will always pick the vertex with the largest d value since the d value is the transmission rate for that path. As other paths to vertices are discovered their d value is determined by comparing the transmission rate of the new path and the current transmission rate. If the new path has a greater transmission rate then the d value is updated. The transmission rate of a path is determined by the minimum of the predecessor's d value or the weight of the $(predecessor, discovered\,vertex)$ edge. This is because the transmission rate is determined by the value of the lowest weight edge, so not only does d value represent transmission rate it also is the value of the current lowest weight edge in the path.

To find the optimal $s \to t$ path, when the algorithm is complete we can just follow the p value starting from t. This is because the p value is the predecessor of the vertex. Since s is the source vertex following the p values will always lead back to s, thus giving a $s \to t$ path.

Below are the 5 claims modified:

Claim 1: For every node v and iterations i, j, if $i \leq j$ then $d_i(v) \leq d_j(v)$

Claim 2: If node u is added to R in iteration i the value of $d(u)$ does not change in iteration i

Claim 3: For every node v and iteration i, if $d_i(v) = k \neq 0$ then there is an R_i-path to v of weight k

Claim 4: For every node u, if u is added to R in iteration i and $d_i(u) = 0$ then there is no $s \to u$ path

Claim 5: For every node u and every iteration $i \geq 1$, if u is added to R in iteration i, then $d_i(u) = \delta(u)$