Solutions for Homework Assignment #2

Answer to Question 1.

a. Suppose $n = 4$, $w_1 = w_2 = 1$, $w_3 = w_4 = 2$, and $C = 3$. The proposed algorithm will output the set $\{\{1, 2\}\}$ (since $w_1 + w_2 \leq 3$ and $w_3 + w_4 > 3$), which is not optimal since $\{\{1, 4\}, \{2, 3\}\}$ is feasible and has more pairs.

b. Without loss of generality, assume that $w_1 \leq w_2 \leq \ldots w_n$ i.e., the campers are listed in increasing weight. (We can do this by sorting the campers.) The proof of correctness is based on two observations.

Claim 1. If $w_1 + w_n > C$ then, for every i, $1 \leq i \leq n$, no optimal set contains the pair $\{i, n\}$.

PROOF. Suppose $w_1 + w_n > C$. For every i, $1 \leq i \leq n$, $w_i \geq w_1$, and so $w_i + w_n \geq w_1 + w_n > C$. So the pair $\{i, n\}$ cannot be in any feasible, and therefore in any optimal, set.

Claim 2. If $w_1 + w_n \leq C$ then there is an optimal set that contains the pair $\{1, n\}$. (This is the hint given in the question.)

PROOF. Suppose $w_1 + w_n \leq C$, and let S^* be an optimal set. S^* must contain the pair $\{1, j\}$, for some j, or the pair $\{i, n\}$, for some i, or both. (If it contained neither such pair, we could add $\{1, n\}$ to S^* and obtains a feasible set with more pairs, contradicting that S^* is optimal.) So, there are three cases.

CASE 1. S^* contains the pair $\{1, j\}$, for some j, but does not contain a pair $\{i, n\}$, for any i. In this case, we replace $\{1, j\}$ by $\{1, n\}$ in S^*, That is, we consider the set $T^* = \big(S^* - \{\{1, j\}\}\big) \cup \{\{1, n\}\}$. Clearly, T^*

- is feasible (since every pair other than $\{1, n\}$ satisfies the weight constraint because it is also in S^*, which is optimal and therefore feasible); and $\{1, n\}$ satisfies the weight constraint by assumption),
- has the same number of pairs as the optimal set S^*, and
- contains the pair $\{1, n\}$.

So, T^* is an optimal set that contains $\{1, n\}$, as wanted.

CASE 2. S^* contains the pair $\{i, n\}$, for some i, but does not contain a pair $\{1, j\}$, for any j. This case is similar to Case 1, except that we now obtain T^* from S^* by replacing $\{i, n\}$ by $\{1, n\}$.

CASE 3. S^* contains both the pair $\{1, j\}$, for some j, and the pair $\{i, n\}$, for some i. In this case obtain T^* from S^* by replacing $\{1, j\}$ and $\{i, n\}$ by $\{1, n\}$ and $\{i, j\}$. Both of the pairs added satisfy the weight constraint: $\{1, n\}$ does so by assumption, and $\{i, j\}$ satisfies the weight constraint because $\{i, n\}$ does (since it is in S^*) and $w_j \leq w_n$. Therefore, T^*

- is feasible,
- has the same number of pairs as the optimal set S^*, and
- contains the pair $\{1, n\}$.

So, T^* is an optimal set that contains $\{1, n\}$, as wanted.

Using these two claims, we can now prove that this greedy algorithm is correct by complete induction on the number of campers n. So, the induction hypothesis is that the algorithm is correct for fewer than n campers.

If $n = 0$ or $n = 1$, the optimal solution is clearly the empty set, and this is what the algorithm returns. So, suppose $n \geq 2$. Without loss of generality, suppose the campers are sorted in increasing weight, so $w_1 \leq w_2 \leq \ldots w_n$. There are two cases to consider.

CASE 1. $w_1 + w_n > C$. By Claim 1, there is no optimal set that contains w_n. Therefore an optimal set among campers $1, 2, \ldots, n$ is also an optimal set among campers $1, 2, \ldots, n - 1$, and, by the induction hypothesis, this is what the algorithm returns.

Case 2. $w_1 + w_n \le C$. In this case, the algorithm returns the set $T = T' \cup \big\{\{1, n\}\big\}$, where, by the induction hypothesis, T' is an optimal set of pairs for campers $2, 3, \ldots, n-1$. Clearly, T is a feasible set for campers $1, 2, \ldots, n$.

By Claim 2, there is an optimal set S^* among campers $1, 2, \ldots, n$ that contains the pair $\{1, n\}$. Clearly, $S' = S^* - \big\{\{1, n\}\big\}$ is an optimal set for campers $2, 3, \ldots, n-1$. (Otherwise, there is a feasible set S'' for campers $2, 3, \ldots, n-1$ with more pairs than S'; but then $S'' \cup \big\{\{1, n\}\big\}$ is a feasible set for campers $1, 2, \ldots, n$ with more pairs than S^*, contradicting that S^* is optimal for campers $1, 2, \ldots, n$.) Since T' and S' are both optimal sets for $2, 3, \ldots, n-1$, $|T'| = |S'|$; hence $|T| = |S^*|$.

So, the algorithm returns a feasible set T that has as many pairs as S^*. Therefore the set T of pairs that the algorithm returns, is an optimal set for $1, 2, \ldots, n$.

c. The proposed algorithm does not always return an optimal solution. For example, suppose there are eight campers A, B, C, D, E, F, G, H, with weights 1, 1, 1, 1, 1, 1, 10, 10 respectively, and the capacity of each canoe is 13. The proposed algorithm will return a single quartet $\{A, B, F, G\}$, which is not optimal because we can form two quartets that satisfy the weight constraint, for example, $\{A, B, C, G\}$ and $\{D, E, F, H\}$.

Answer to Question 2.

In Dijkstra's algorithm we keep track of a set R of nodes, and for each node v two quantities, $d(v)$ and $pre(v)$: $d(v)$ is the minimum length of any $s \to v$ path among all paths whose intermediate nodes (i.e., nodes other than v) are in R, or ∞ if no such path exists; and $pre(v)$ is the predecessor of v on such a path. In each iteration we augment R by a node u, currently not in S, that has minimum $d(u)$, and we update $d(v)$ and $pre(v)$ for each node v adjacent to u to reflect the change in R.

In our modification of Dijkstra's algorithm, instead of $d(v)$ we keep track of $tr(v)$: the maximum transmission rate of any $s \to v$ path among all paths whose intermediate nodes are in R, or 0 if no such path exists. In each iteration we augment R by a node u, currently not in R, that has ***maximum*** $tr(u)$, and we update $tr(v)$ and $pre(v)$. When R contains all nodes, $tr(v)$ is the weight of an $s \to v$ path of maximum transmission rate. Starting with t and using the computed values of pre we find (in reverse) an $s \to t$ path P of maximum transmission rate, and return it. The algorithm is shown below:

MaxRate($G, \mathbf{wt}, s, t$)
$R := \varnothing$
$tr(s) := \infty$; $pre(s) :=$ NIL
for each $v \in V - \{s\}$ **do** $tr(v) := 0$; $pre(v) :=$ NIL
while $R \neq V$ **do**
  let u be a node ***not*** in R with maximum tr-value (i.e., $u \in V - R$ and $\forall\, u' \in V - R, tr(u) \ge tr(u')$)
  $R := R \cup \{u\}$
  for each $v \in V$ such that $(u, v) \in E$ **do**
    if $\min\big(tr(u), \mathbf{wt}(u, v)\big) > tr(v)$ **then** $tr(v) := \min\big(tr(u), \mathbf{wt}(u, v)\big)$; $pre(v) := u$
$P :=$ empty sequence
$u := t$
while $u \neq$ NIL **do**
  prepend u to P; $u := pre(u)$
return P

The five claims in the proof of correctness of Dijkstra's algorithm now become as follows. Let $\tau(v)$ be the maximum transmission rate of any $s \to v$ path, or 0, if no such path exists.

Claim 1. *For every node v and iterations i, j, if $i \le j$ then $tr_i(v) \le tr_j(v)$.*

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

Claim 3. *For every node v and iteration i, if $tr_i(v) = k \neq 0$ 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 $tr_i(u) = 0$ then there is no $s \to u$ path.*

Claim 5. *For every node u and every iteration $i \ge 1$, if u is added to R in iteration i, then $tr_i(u) = \tau(u)$.*