Test 1 Review
CSCI 432

Test 1 Logistics

1. During class on Thursday 2/13.
2. You can bring your book and any notes you would like, but no electronic devices.
3. You may assume anything proven in class or on homework.
4. Four questions (15 points):
    1) Identify optimal substructure equation (5 points).
    2) Write pseudocode for provided algorithm (3 points).
    3) Argue validity of provided algorithm (2 points).
    4) Argue optimality of provided algorithm (5 points).

Test 1 Logistics

1. During class on Thursday 2/13.
2. You can bring your book and any notes you would like, but no electronic devices.
3. You may assume anything proven in class or on homework.
4. Four questions (15 points):
   1) Identify optimal substructure equation (5 points).
   2) Write pseudocode for provided algorithm (3 points).
   3) Argue validity of provided algorithm (2 points).
   4) Argue optimality of provided algorithm (5 points).

Making Change

In general, suppose a country has denominations:

$$1 = d_1 < d_2 < \cdots < d_k$$

(US coins: $d_1 = 1, d_2 = 5, d_3 = 10, d_4 = 25$)

$C(p)$ – minimum number of coins to make p cents.

x – value (e.g. \$0.25) of a coin used in the optimal solution.

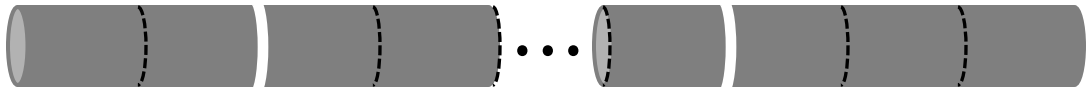
So, we'll check all possibilities.

But we don't know what actual coin is in the optimal solution.

$$C(p) = \begin{cases} \min\limits_{d_i \le p} C(p - d_i) + 1 \,, p > 0 \\ 0 \quad , p = 0 \end{cases}$$

Rod Cutting

length	1	2	3	4	…	n
profit	\$1	\$5	\$8	\$9	…	\$?



O_n = optimal profit from partitioning rod of length n.
p_i = profit for rod of length i.

$$O_n = \max_{1 \le i \le n} (p_i + O_{n-i})$$

Weighted Graph Problem

Input: n node graph, $G = (V, E)$, with vertex weights.

Output: Subset of vertices such that they do not share any edges with each other, and the sum of their weights (w_i) is maximized.

Optimal solution to V_i is either the optimal for V_{i-1} (with vertex i discarded), or the optimal for V_{i-2} (with vertex i included).

Let $A[i]$ be the weight of the optimal solution for V_i.

$$A[i] = \max \begin{pmatrix} A[i-1] \\ A[i-2] + w_i \end{pmatrix}$$

Edit Distance

We want to align two strings, $x = [x_1, \dots, x_n]$ and $y = [y_1, \dots, y_m]$.

$E(i, j)$ = optimal cost of aligning $[x_1, \dots, x_i]$ and $[y_1, \dots, y_j]$.

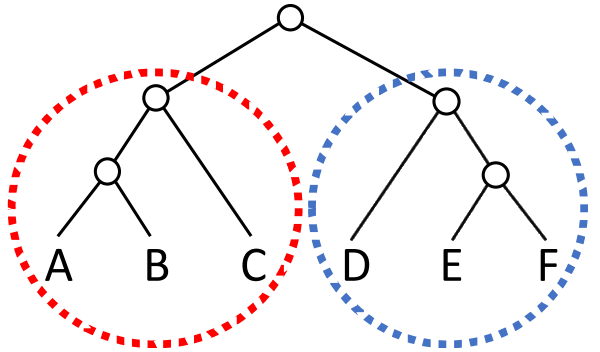
Can we say anything about optimal alignment of $[x_1, \dots, x_i]$ and $[y_1, \dots, y_j]$?

Alignment	Cost
$\begin{matrix} x_i \\ y_j \end{matrix}$	$\{0,1\} + E(i-1, j-1)$
$\begin{matrix} x_i \\ - \end{matrix}$	$1 + E(i-1, j)$
$\begin{matrix} - \\ y_j \end{matrix}$	$1 + E(i, j-1)$

$$E(i,j) = \min \begin{cases} \text{diff}(i,j) + E(i-1, j-1) \\ 1 + E(i-1, j) \\ 1 + E(i, j-1) \end{cases}$$

$$\text{where diff}(i,j) = \begin{cases} 0, & x_i = y_j \\ 1, & x_i \neq y_j \end{cases}$$

Matrix-Chain Multiplication



Input: n matrices $M_1, M_2, \dots, M_n$ with dimensions of:

$$m_0 \times m_1,\ m_1 \times m_2,\ \dots,\ m_{n-1} \times m_n$$

Output: Minimum number of scalar multiplications needed to calculate $M_1 \times M_2 \times \cdots \times M_n$.

Let $C[i,j]$ = Minimum number of scalar multiplications needed for multiplying $M_i \times M_{i+1} \times \cdots \times M_j$

Suppose the final multiplication in the optimal solution occurs at k:

$$(M_i \times \cdots \times M_k)(M_{k+1} \times \cdots \times M_j)$$

$$\boldsymbol{C[i,j] = \min_{i \le k < j}\left(C[i,k] + C[k+1,j] + m_k \times m_{i-1} \times m_j\right)}$$

Test 1 Logistics

1. During class on Thursday 2/13.
2. You can bring your book and any notes you would like, but no electronic devices.
3. You may assume anything proven in class or on homework.
4. Four questions (15 points):
   1) Identify optimal substructure equation (5 points).
   2) Write pseudocode for provided algorithm (3 points).
   3) Argue validity of provided algorithm (2 points).
   4) Argue optimality of provided algorithm (5 points).

Single Room Scheduling

```
room_scheduling(courses C)
   C.sort_by_finish()
   selected = {C[1]}
   last_added = 1
   for i = 2 to C.length
      if C[i].start ≥ C[last_added].finish
         selected.add(C[i])
         last_added = i
   return selected
```

Test 1 Logistics

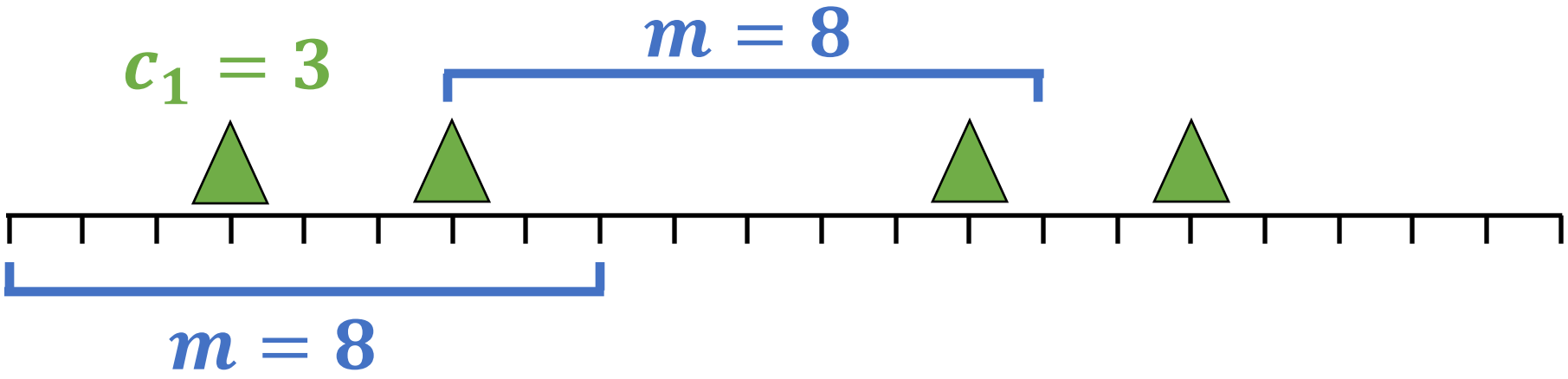
1. During class on Thursday 2/13.
2. You can bring your book and any notes you would like, but no electronic devices.
3. You may assume anything proven in class or on homework.
4. Four questions (15 points):
   1) Identify optimal substructure equation (5 points).
   2) Write pseudocode for provided algorithm (3 points).
   3) Argue validity of provided algorithm (2 points).
   4) Argue optimality of provided algorithm (5 points).

Test 1 Logistics

1. During class on Thursday 2/13.
2. You can bring your book and any notes you would like, but no electronic devices.
3. You may assume anything proven in class or on homework.
4. Four questions (15 points):
    1) Identify optimal substructure equation (5 points).
    2) Write pseudocode for provided algorithm (3 points).
    3) Argue validity of provided algorithm (2 points).
    4) Argue optimality of provided algorithm (5 points).

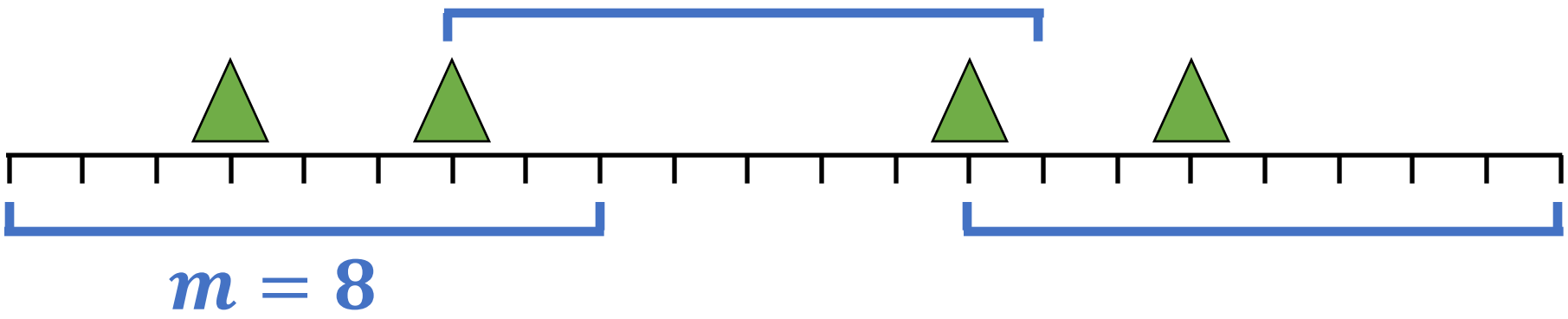
Problem Statement

Suppose you are going on a long backpacking trip on some trail. You can hike up to m miles a day. There are designated campsites along the trail at miles $(c_1, c_2, \ldots, c_n)$. You need to stay in a designated campsite each night. Your goal is to hike the whole trail in as few days as possible.



Greedy Algorithm

Greedy Campsite Selection (GCS): Camp at the last reachable (i.e. $< m$) campsite each day.



Greedy Algorithm

Theorem: Camping at the last reachable campsite each day is optimal.

Proof:

Prove this by arguing that you can swap S_{ALG} campsites into S_{OPT} and keep S_{OPT} both valid and optimal.

Theorem: Camping at the last reachable campsite each day is optimal.

Proof: Let $S_{ALG} = (c_{a_1}, \dots, c_{a_k})$ be the campsites chosen by GCS and $S_{OPT} = (c_{o_1}, \dots, c_{o_t})$ be an optimal selection.

Suppose night i is the first night that S_{ALG} and S_{OPT} do not stop at the same site.

Consider $S'_{OPT} = S_{OPT} \setminus \{c_{o_i}\} \cup \{c_{a_i}\}$.

c_{a_i} will not require S_{OPT} to go farther than m on day i, since it worked in S_{ALG} and $c_{a_{i-1}} = c_{o_{i-1}}$. Likewise, c_{a_i} will also work for $c_{o_{i+1}}$ since $c_{o_i} \leq c_{a_i}$, so if $c_{o_{i+1}} - c_{o_i} \leq m$ then $c_{o_{i+1}} - c_{a_i} \leq m$ too.

We can repeat this by replacing campsites in S_{OPT} with the corresponding site in S_{ALG} without exceeding the m-restriction or increasing the number of sites.

Thus, S_{ALG} is optimal.

Single Room Scheduling

Greedy decision: Select the next course with the earliest compatible finish time.

Proof of optimality: Let C be the set of courses, $S_{ALG} \subseteq C$ be the greedy algorithm's selection, and $S_{OPT} \subseteq C$ be an optimal selection, all sorted by increasing finish time.

Suppose $S_{ALG}[i] = S_{OPT}[i]$, for all $i < k$ and $S_{ALG}[k] = c_i \neq c_j = S_{OPT}[k]$.

Create the revised schedule $S'_{OPT} = S_{OPT} \setminus \{c_j\} \cup \{c_i\}$. (I.e., Swap $S_{ALG}[k]$ for $S_{OPT}[k]$)

c_i is compatible with previous courses in S'_{OPT} since $S_{ALG}[i] = S_{OPT}[i] = S'_{OPT}[i]$, for all $i < k$

c_i is compatible with subsequent courses in S'_{OPT} since $f_i \leq f_j$. Otherwise, the greedy algorithm would have selected c_j instead of c_i.

So S'_{OPT} is a valid schedule with the same number of courses as S_{OPT}, so S'_{OPT} is also optimal.

We can then proceed inductively and show that each course in S_{OPT} can be replaced by the corresponding course in S_{ALG} without violating compatibility. Since replacing every course in S_{OPT} with the courses in S_{ALG} keeps the solution optimal, S_{ALG} must be optimal. (i.e., we translated S_{OPT} into S_{ALG} at no extra cost).

Greedy Change Making (USD)

Greedy decision: Select the highest denomination coin that evenly subtracts from the remaining balance (i.e. max quarters + max dimes + max nickels + max pennies).

Proof of optimality: Let $V =$ \$ value. Let q, d, n, p be the number of quarters, dimes, nickels, and pennies used.

Let $T_{ALG} = q_{ALG} + d_{ALG} + n_{ALG} + p_{ALG}$ be the algorithm's solution and $T_{OPT} = q_{OPT} + d_{OPT} + n_{OPT} + p_{OPT}$ be an optimal solution.

$q_{OPT} \leq q_{ALG}$ since q_{ALG} is the largest value such that $V - 25\, q_{ALG} < 25$

If $q_{OPT} < q_{ALG}$, $V - 25\, q_{OPT} \geq 25$.

I.e. $V - 25\, q_{OPT} = 10\, d_{OPT} + 5\, n_{OPT} + p_{OPT}$

$= 25 + 10(d_{OPT} - 2) + 5(n_{OPT} - 1) + p_{OPT}$

$\Rightarrow V = 25(q_{OPT} + 1) + 10(d_{OPT} - 2) + 5(n_{OPT} - 1) + p_{OPT}$

$\Rightarrow$ Num coins $= q_{OPT} + d_{OPT} + n_{OPT} + p_{OPT} - 2 < T_{OPT}$. Contradiction.

Thus, $q_{OPT} = q_{ALG}$.

Repeat argument to show that $d_{OPT} = d_{ALG}$, $n_{OPT} = n_{ALG}$, and $p_{OPT} = p_{ALG}$. Thus, $T_{OPT} = T_{ALG}$ and T_{ALG} is optimal.

Optimality

1. ?

Optimality

1. Consider the output from the algorithm.
2. ?

Optimality

1. Consider the output from the algorithm.
2. Consider an optimal solution.
3. ?

Optimality

1. Consider the output from the algorithm.
2. Consider an optimal solution.
3. Go to where OPT and ALG disagree.
4. ?

Optimality

1. Consider the output from the algorithm.
2. Consider an optimal solution.
3. Go to where OPT and ALG disagree.
4. Put ALG's choice into OPT and show:
    a. ?
    b. ?

Optimality

1. Consider the output from the algorithm.
2. Consider an optimal solution.
3. Go to where OPT and ALG disagree.
4. Put ALG's choice into OPT and show:
    a. The result is still valid.
    b. The cost is the same.
5. ?

Optimality

1. Consider the output from the algorithm.
2. Consider an optimal solution.
3. Go to where OPT and ALG disagree.
4. Put ALG's choice into OPT and show:
    a. The result is still valid.
    b. The cost is the same.
5. Repeat for all other disagreements.
6. ?

Optimality

1. Consider the output from the algorithm.
2. Consider an optimal solution.
3. Go to where OPT and ALG disagree.
4. Put ALG's choice into OPT and show:
    a. The result is still valid.
    b. The cost is the same.
5. Repeat for all other disagreements.
6. Implies ALG = OPT.

Greedy Algorithm

Theorem: Camping at the last reachable campsite each day is optimal.

Proof:

Prove this by arguing that you can swap S_{ALG} campsites into S_{OPT} and keep S_{OPT} both valid and optimal.

Theorem: Camping at the last reachable campsite each day is optimal.

Proof: Let $S_{ALG} = (c_{a_1}, \ldots, c_{a_k})$ be the campsites chosen by GCS and $S_{OPT} = (c_{o_1}, \ldots, c_{o_t})$ be an optimal selection.

Prove this by arguing that $c_{o_i} \leq c_{a_i}$ for every single night i.

Theorem: Camping at the last reachable campsite each day is optimal.

Proof: Let $S_{ALG} = (c_{a_1}, \ldots, c_{a_k})$ be the campsites chosen by GCS and $S_{OPT} = (c_{o_1}, \ldots, c_{o_t})$ be an optimal selection.

First show that $c_{o_i} \leq c_{a_i}$ for each $i = 1,2, \ldots$ using induction.

Theorem: Camping at the last reachable campsite each day is optimal.

Proof: Let $S_{ALG} = (c_{a_1}, \ldots, c_{a_k})$ be the campsites chosen by GCS and $S_{OPT} = (c_{o_1}, \ldots, c_{o_t})$ be an optimal selection.

First show that $c_{o_i} \leq c_{a_i}$ for each $i = 1,2, \ldots$ using induction.

I.e., Show:

1. **On night 1 that $c_{o_1} \leq c_{a_1}$.**
2. **If $c_{o_{i-1}} \leq c_{a_{i-1}}$ on night $i-1$, then $c_{o_i} \leq c_{a_i}$ on night i.**

Theorem: Camping at the last reachable campsite each day is optimal.

Proof: Let $S_{ALG} = (c_{a_1}, \ldots, c_{a_k})$ be the campsites chosen by GCS and $S_{OPT} = (c_{o_1}, \ldots, c_{o_t})$ be an optimal selection.

First show that $c_{o_i} \leq c_{a_i}$ for each $i = 1,2, \ldots$ using induction. Clearly $c_{o_1} \leq c_{a_1}$ because the algorithm picks the farthest site on night 1.

Theorem: Camping at the last reachable campsite each day is optimal.

Proof: Let $S_{ALG} = (c_{a_1}, \ldots, c_{a_k})$ be the campsites chosen by GCS and $S_{OPT} = (c_{o_1}, \ldots, c_{o_t})$ be an optimal selection.

First show that $c_{o_i} \leq c_{a_i}$ for each $i = 1,2, \ldots$ using induction. Clearly $c_{o_1} \leq c_{a_1}$ because the algorithm picks the farthest site on night 1.

Suppose $c_{o_{i-1}} \leq c_{a_{i-1}}$ for some $i - 1$.

Theorem: Camping at the last reachable campsite each day is optimal.

Proof: Let $S_{ALG} = (c_{a_1}, \ldots, c_{a_k})$ be the campsites chosen by GCS and $S_{OPT} = (c_{o_1}, \ldots, c_{o_t})$ be an optimal selection.

First show that $c_{o_i} \leq c_{a_i}$ for each $i = 1,2, \ldots$ using induction. Clearly $c_{o_1} \leq c_{a_1}$ because the algorithm picks the farthest site on night 1.

Suppose $c_{o_{i-1}} \leq c_{a_{i-1}}$ for some $i - 1$. Then, $c_{o_i} - c_{o_{i-1}} \leq m$ (can't go farther than m in a day)

Theorem: Camping at the last reachable campsite each day is optimal.

Proof: Let $S_{ALG} = (c_{a_1}, \ldots, c_{a_k})$ be the campsites chosen by GCS and $S_{OPT} = (c_{o_1}, \ldots, c_{o_t})$ be an optimal selection.

First show that $c_{o_i} \leq c_{a_i}$ for each $i = 1,2, \ldots$ using induction. Clearly $c_{o_1} \leq c_{a_1}$ because the algorithm picks the farthest site on night 1.

Suppose $c_{o_{i-1}} \leq c_{a_{i-1}}$ for some $i - 1$. Then, $c_{o_i} - c_{o_{i-1}} \leq m$ (can't go farther than m in a day) and $c_{o_i} - c_{a_{i-1}} \leq m$.

$$\boldsymbol{c_{o_i} - c_{a_{i-1}} \leq c_{o_i} - c_{o_{i-1}} \leq m}$$

$$\boldsymbol{c_{o_{i-1}} \leq c_{a_{i-1}}}$$

Subtracting a bigger number makes the result smaller.

Theorem: Camping at the last reachable campsite each day is optimal.

Proof: Let $S_{ALG} = (c_{a_1}, \ldots, c_{a_k})$ be the campsites chosen by GCS and $S_{OPT} = (c_{o_1}, \ldots, c_{o_t})$ be an optimal selection.

First show that $c_{o_i} \leq c_{a_i}$ for each $i = 1,2, \ldots$ using induction. Clearly $c_{o_1} \leq c_{a_1}$ because the algorithm picks the farthest site on night 1.

Suppose $c_{o_{i-1}} \leq c_{a_{i-1}}$ for some $i - 1$. Then, $c_{o_i} - c_{o_{i-1}} \leq m$ (can't go farther than m in a day) and $c_{o_i} - c_{a_{i-1}} \leq m$. This means that on day i, S_{ALG} could stop at c_{o_i} since it is within m of $c_{a_{i-1}}$.

Theorem: Camping at the last reachable campsite each day is optimal.

Proof: Let $S_{ALG} = (c_{a_1}, \dots, c_{a_k})$ be the campsites chosen by GCS and $S_{OPT} = (c_{o_1}, \dots, c_{o_t})$ be an optimal selection.

First show that $c_{o_i} \leq c_{a_i}$ for each $i = 1,2, \dots$ using induction. Clearly $c_{o_1} \leq c_{a_1}$ because the algorithm picks the farthest site on night 1.

Suppose $c_{o_{i-1}} \leq c_{a_{i-1}}$ for some $i - 1$. Then, $c_{o_i} - c_{o_{i-1}} \leq m$ (can't go farther than m in a day) and $c_{o_i} - c_{a_{i-1}} \leq m$. This means that on day i, S_{ALG} could stop at c_{o_i} since it is within m of $c_{a_{i-1}}$. It could also maybe go farther. Thus, $c_{o_i} \leq c_{a_i}$.

Theorem: Camping at the last reachable campsite each day is optimal.

Proof: Let $S_{ALG} = (c_{a_1}, \ldots, c_{a_k})$ be the campsites chosen by GCS and $S_{OPT} = (c_{o_1}, \ldots, c_{o_t})$ be an optimal selection.

First show that $c_{o_i} \leq c_{a_i}$ for each $i = 1,2, \ldots$ using induction. Clearly $c_{o_1} \leq c_{a_1}$ because the algorithm picks the farthest site on night 1.

Suppose $c_{o_{i-1}} \leq c_{a_{i-1}}$ for some $i - 1$. Then, $c_{o_i} - c_{o_{i-1}} \leq m$ (can't go farther than m in a day) and $c_{o_i} - c_{a_{i-1}} \leq m$. This means that on day i, S_{ALG} could stop at c_{o_i} since it is within m of $c_{a_{i-1}}$. It could also maybe go farther. Thus, $c_{o_i} \leq c_{a_i}$.

Now argue that $k = t$.

Theorem: Camping at the last reachable campsite each day is optimal.

Proof: Let $S_{ALG} = (c_{a_1}, \ldots, c_{a_k})$ be the campsites chosen by GCS and $S_{OPT} = (c_{o_1}, \ldots, c_{o_t})$ be an optimal selection.

First show that $c_{o_i} \leq c_{a_i}$ for each $i = 1,2, \ldots$ using induction. Clearly $c_{o_1} \leq c_{a_1}$ because the algorithm picks the farthest site on night 1.

Suppose $c_{o_{i-1}} \leq c_{a_{i-1}}$ for some $i - 1$. Then, $c_{o_i} - c_{o_{i-1}} \leq m$ (can't go farther than m in a day) and $c_{o_i} - c_{a_{i-1}} \leq m$. This means that on day i, S_{ALG} could stop at c_{o_i} since it is within m of $c_{a_{i-1}}$. It could also maybe go farther. Thus, $c_{o_i} \leq c_{a_i}$.

We know that S_{ALG} cannot finish before S_{OPT}, since S_{OPT} is optimal.

Theorem: Camping at the last reachable campsite each day is optimal.

Proof: Let $S_{ALG} = (c_{a_1}, \ldots, c_{a_k})$ be the campsites chosen by GCS and $S_{OPT} = (c_{o_1}, \ldots, c_{o_t})$ be an optimal selection.

First show that $c_{o_i} \leq c_{a_i}$ for each $i = 1,2, \ldots$ using induction. Clearly $c_{o_1} \leq c_{a_1}$ because the algorithm picks the farthest site on night 1.

Suppose $c_{o_{i-1}} \leq c_{a_{i-1}}$ for some $i - 1$. Then, $c_{o_i} - c_{o_{i-1}} \leq m$ (can't go farther than m in a day) and $c_{o_i} - c_{a_{i-1}} \leq m$. This means that on day i, S_{ALG} could stop at c_{o_i} since it is within m of $c_{a_{i-1}}$. It could also maybe go farther. Thus, $c_{o_i} \leq c_{a_i}$.

We know that S_{ALG} cannot finish before S_{OPT}, since S_{OPT} is optimal. So, on S_{OPT}'s final day, $c_{o_t} \leq c_{a_t}$

Theorem: Camping at the last reachable campsite each day is optimal.

Proof: Let $S_{ALG} = (c_{a_1}, \ldots, c_{a_k})$ be the campsites chosen by GCS and $S_{OPT} = (c_{o_1}, \ldots, c_{o_t})$ be an optimal selection.

First show that $c_{o_i} \leq c_{a_i}$ for each $i = 1,2, \ldots$ using induction. Clearly $c_{o_1} \leq c_{a_1}$ because the algorithm picks the farthest site on night 1.

Suppose $c_{o_{i-1}} \leq c_{a_{i-1}}$ for some $i - 1$. Then, $c_{o_i} - c_{o_{i-1}} \leq m$ (can't go farther than m in a day) and $c_{o_i} - c_{a_{i-1}} \leq m$. This means that on day i, S_{ALG} could stop at c_{o_i} since it is within m of $c_{a_{i-1}}$. It could also maybe go farther. Thus, $c_{o_i} \leq c_{a_i}$.

We know that S_{ALG} cannot finish before S_{OPT}, since S_{OPT} is optimal. So, on S_{OPT}'s final day, $c_{o_t} \leq c_{a_t}$, which means that S_{ALG} is also done on that day

Theorem: Camping at the last reachable campsite each day is optimal.

Proof: Let $S_{ALG} = (c_{a_1}, \dots, c_{a_k})$ be the campsites chosen by GCS and $S_{OPT} = (c_{o_1}, \dots, c_{o_t})$ be an optimal selection.

First show that $c_{o_i} \leq c_{a_i}$ for each $i = 1,2, \dots$ using induction. Clearly $c_{o_1} \leq c_{a_1}$ because the algorithm picks the farthest site on night 1.

Suppose $c_{o_{i-1}} \leq c_{a_{i-1}}$ for some $i - 1$. Then, $c_{o_i} - c_{o_{i-1}} \leq m$ (can't go farther than m in a day) and $c_{o_i} - c_{a_{i-1}} \leq m$. This means that on day i, S_{ALG} could stop at c_{o_i} since it is within m of $c_{a_{i-1}}$. It could also maybe go farther. Thus, $c_{o_i} \leq c_{a_i}$.

We know that S_{ALG} cannot finish before S_{OPT}, since S_{OPT} is optimal. So, on S_{OPT}'s final day, $c_{o_t} \leq c_{a_t}$, which means that S_{ALG} is also done on that day, so $k = t$ and S_{ALG} is optimal.