Problem 1. Consider the following problem, Counting Subset Sums:

input: A pair $I = (v, T)$ where $v = (v_1, v_2, \ldots, v_n)$ is a sequence of positive integers and $T \geq 0$ is an integer.

output: The number of subsequences of v with total T. That is,

$$\Big|\big\{S \subseteq \{1, 2, \ldots, n\} : v(S) = T\big\}\Big|,$$

where for notational convenience throughout we define $v(S) = \sum_{i \in S} v_i$.

This problem differs from Counting Ways to Make Change/ In this problem each of the given numbers can be used at most once.

Fix an input (v, T). For each $i \in \{0, 1, \ldots, n\}$ and $t \in \{0, 1, \ldots, T\}$, define $N(i, t)$ to be the number of subsequences of $(v_1, \ldots, v_i)$ with total t. That is,

$$N(i, t) = \Big|\big\{S \subseteq \{1, 2, \ldots, i\} : v(S) = t\big\}\Big|.$$

The correct output for the given input is $N(n, T)$.

(a) Give a recurrence relation for $N(i, t)$ for all $i \in \{0, 1, \ldots, n\}$ and $t \in \{0, 1, \ldots, T\}$, including the base cases. State the recurrence relation as a lemma (see the template).

(b) Complete the partial long-form proof of the lemma given in the template. You can use a proof structure similar to the one for the long-form proof of correctness for Knapsack.

You don't need to give the resulting algorithm or analyze the running time.

Problem 1 answer

(a)

Lemma 1. *For any $i \in \{0, 1, \ldots, n\}$ and $t \in \{0, 1, \ldots, T\}$, for N as defined in the problem statement,*

$$N(i,t) = \begin{cases} 1 & \text{if } t = 0 \text{ and } i = 0, \\ 0 & \text{if } t > 0 \text{ and } i = 0, \\ N(i-1,t) + N(i-1,t-x_i) & \text{if } i > 0 \text{ and } v_i \leq t, \\ N(i-1,t) & \text{if } i > 0 \text{ and } v_i > t. \end{cases}$$

(b)

Proof (long form).

1. Consider any $i \in \{0, \ldots, n\}$ and $t \in \{0, 1, \ldots, T\}$.
2. Recall that $N(i,t) = \left|\{S \subseteq \{1, 2, \ldots, i\} : \sum_{i \in S} v_i = t\}\right|$.

3.1. Case 1. Consider the first base case $t = 0$ and $i = 0$.
3.2. The only sub-sequence of the empty set that has total 0 is itself.
3.3. So $N(i,t) = 1$.

4.1. Case 2. Consider the second base case $t > 0$ and $i = 0$.
4.2. There is no way to make change for a nonzero total t with the empty set, which has total 0.
4.3. So $N(i,t) = 0$.

5.1. Case 3. Next consider the case when $i > 0$ and $v_i \leq t$.
5.2. Consider the subsets $S \subseteq \{1, 2, \ldots, i\}$ such that $v(S) = t$.
5.3. Partition these into two types: (A) those that don't include i, and (B) those that do.
5.4. $N(i,t)$ is the number of subsets of either type.

5.5. The type-A subsets are just the subsets of $\{1, \ldots, i-1\}$ with $v(S) = t$.
5.6. Thus, the total number of type-A subsets is just $N(i-1,t)$.

5.7. The type-B subsets are the subsets where v_i is used. These subsets are the subsets of $\{1, \ldots, i-1\}$ with $v(S) = t - v_i$.
5.8. Thus, the total number of type-B subsets is just $N(i-1, t-v_i)$.
5.9. By Steps 5.4, 5.6 and 5.8, $N(i,t) = N(i-1,t) + N(i-1,t-v_i)$.
5.10. So the recurrence holds in this case.

6.1. Case 4. Finally consider the remaining case, when $i > 0$ and $v_i > t$.
6.2. Since $v_i > t$, we cannot have a sub-sequence that uses v_i and has a total of t.
6.3. We would exclude v_i from the sub-sequence we are counting. These subsets are the subsets of $\{1, \ldots, i-1\}$ with $v(S) = t$.
6.4. So $N(i,t) = N(i-1,t)$.
6.5. So the recurrence holds in this case.

7. By the case analysis (Cases 1–4) above, the recurrence holds. $\square$

Problem 2. Design a dynamic-programming algorithm for Smallest Subset Sum:

input: A pair $I = (v, T)$ where $v = (v_1, v_2, \ldots, v_n)$ is a sequence of positive integers and $T \geq 0$ is an integer.

output: The minimum size of any subsequence of v with total T:

$$\min \big\{ |S| : S \subseteq \{1, 2, \ldots, n\},\, v(S) = T \big\},$$

where, for notational convenience throughout we define $v(S) = \sum_{i \in S} v_i$.

Recall that the minimum of the empty set is ∞. (So, if no subsequence of v has total T, the output should be ∞.)

(a) Define the subproblems that your algorithm solves for a given input $(v = (v_1, \ldots, v_n), T)$.

(b) Which subproblem gives the final answer?

(c) State an appropriate recurrence relation, including boundary cases.

(d) In terms of n and T, what is the big-Θ running time of the resulting iterative (or recursive and memoized) algorithm?

Problem 2 answer

(a) Here are the subproblems that the algorithm solves.
For each $i \in \{0, 1, \ldots, n\}$ and $t \in \{0, 1, \ldots, T\}$, define $N(i, t)$ to be the minimum size sub-sequence of $(v_1, \ldots, v_i)$ with total t. That is,

$$N(i,t) = \min\left\{|S| : S \subseteq \{1, 2, \ldots, i\},\ v(S) = t\right\}$$

(b) The correct output for the given input is

$$N(n, T)$$

(c) Here is the recurrence relation:

$$N(i,t) = \begin{cases} 0 & \text{if } t = 0 \text{ and } i = 0, \\ \infty & \text{if } t > 0 \text{ and } i = 0, \\ \min\left\{N(i-1, t), N(i-1, t-x_i) + 1\right\} & \text{if } i > 0 \text{ and } v_i \leq t, \\ N(i-1, t) & \text{if } i > 0 \text{ and } v_i > t. \end{cases}$$

(d) The running time is

$$\Theta(nT)$$

Problem 3. Design an $O(\ell mn)$-time dynamic-programming algorithm for the Three-Way LCS problem:

input: a triple $I = (A[1..\ell], B[1..m], C[1..n])$ of three sequences.

output: the maximum length of any common subsequence of $A[1..\ell]$, of $B[1..m]$, and of $C[1..n]$. (That is, the maximum length of any sequence that is simultaneously a subsequence of A, of B, and of C.)

(a) Define the subproblems the algorithm solves given input $(A[1..\ell], B[1..m], C[1..n])$.

(b) Which subproblem gives the final answer?

(c) State the recurrence relation, including boundary cases.

(d) What is the running time of the resulting (iterative, or recursive and memoized) algorithm?

Problem 3 answer

(a) Here are the subproblems that the algorithm solves.
For each $i \in \{0, 1, \ldots, \ell\}$, $j \in \{0, 1, \ldots, m\}$, and $k \in \{0, 1, \ldots, n\}$, define S(i,j,k) to be the length of the longest common sub-sequence between A[1..i], B[1..j], and C[1..k].

(b) The correct output for the given input is

$$S(l, m, n)$$

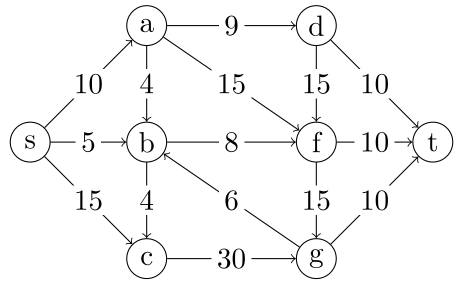
(c) Here is the recurrence relation:

$$S(i,j,k) = \begin{cases} 0 & \text{if } i = 0 \text{ or } j = 0 \text{ or } k = 0, \\ 1 + S(i-1, j-1, k-1) & \text{if } A[i] = B[j] = C[k], \\ \max\left\{S(i-1,j,k), S(i,j-1,k), S(i,j,k-1)\right\} & \text{if } A[i] \neq B[j] \text{ or } A[i] \neq C[k]. \end{cases}$$

(d) The running time is

$$\Theta(lmn)$$

Problem 4. Consider the flow network $G = (V, E)$ with edge capacities as shown below:



(a) Find a maximum-value s-t flow f in the network. Draw the network, labeling each edge (u, w) with the flow $f(u, w)$ on the edge.

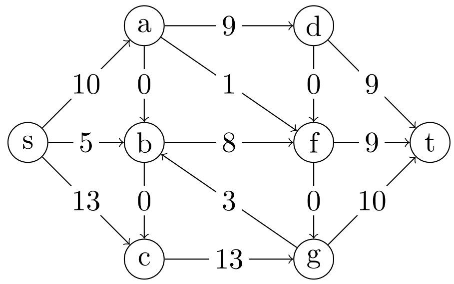
(b) What is the value of the flow?

(c) Find an s-t cut $(S, T = S \setminus V)$ of minimum capacity in the network. For the cut you found, what is S?

(d) And what is the capacity of the cut?

Problem 4 answer

(a) Here is the flow:



(b) The value of the flow is

$$28$$

(c) The s-t cut is $(S, T = V \setminus S)$ where

$$S = \{s, b, c, g\}.$$

(d) The capacity of the cut is

$$28$$