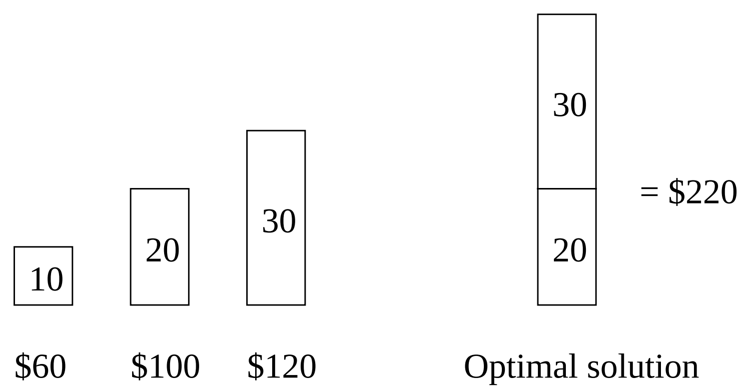
CPS 130 Homework 15 - Solutions

1. In this problem we consider the 0-1 KNAPSACK PROBLEM: Given n items, with item i being worth $v[i]$ dollars and having weight $w[i]$ pounds, fill a knapsack of capacity m pounds with the maximal possible value.

   Example: Given a knapsack of capacity 50, the maximal value obtainable with three items of value \$60, \$100, and \$120 and weights 10, 20, and 30, respectively, is \$220.

   The algorithm `Knapsack(i,j)` below returns the maximal value obtainable when filling a knapsack of capacity j using items among items 1 through i (`Knapsack(n,m)` solves our problem). The algorithm works by recursively computing the best solution obtainable *with* the last item and the best solution obtainable *without* the last item, and returning the best of them.

```
Knapsack(i,j)

  IF w[i] <= j THEN
    with = v[i] + Knapsack(i-1, j-w[i])
  ELSE
    with = 0
  END IF
  without = Knapsack(i-1,j)
  RETURN max{with, without}

END
```

   (a) Show that the running time T of `Knapsack`(n, m) is exponential in n or m. (*Hint:* look at the case where $w[i] = 1$ for all $1 \leq i \leq n$ and show that $T(n, m) = \Omega(2^{\min(m,n)})$).

(b) Describe an $O(n \cdot m)$ algorithm for computing the value of the optimal solution.

Solution:

(a) Following the hint, if $w[i] = 1$ then it is clear that $T(n, m) > 2T(n - 1, m - 1) + 1$. This recurrence, which runs for $\min(m, n)$ steps, gives that $T(n, m) = \Omega(2^{\min(m,n)})$.

(b) We create a table of size $[n][m]$ in which to store our results of prior runs. The modified algorithm would be as follows:

```
Knapsack(i,j)

  IF table[i][j] != 0 THEN
    RETURN table[i][j]
  IF w[i] <= j THEN
    with = v[i] + Knapsack(i-1, j-w[i])
  ELSE
    with = 0
  without = Knapsack(i-1,j)
  table[i][j] = max{with, without}
  RETURN max{with, without}

END
```

This will run in $O(n \cdot m)$ time as we fill each entry in the table at most once, and there are nm spaces in the table.