CS3230 Assignment 8

zhuang Jianning
A0214561M

1. variant of knapsack problem with extra parameter $R \in \{1, 2, .. n\}$ which lets you choose at most R items out of n items with total weight at most W. Maximize value of items.

Let $m[i, j, k]$ be the maximum value obtained using:

a subset of items in $\{1, 2, .. i\}$

with total weight $\leq j$

using at most k items

$$m[i,j,k] = \begin{cases} 0, & \text{if } i=0 \text{ or } j=0 \text{ or } k=0 \\ \max\{ m[i-1, j-w_i, k-1] + v_i \ , \ m[i-1, j, k] \}, & \text{if } w_i \leq j \\ m[i-1, j, k], & \text{otherwise} \end{cases}$$

```
KNAPSACK (v, w, n, W, R)
    initialize 3D table m[n, W, R] to 0
    for i = 1 to n
        for j = 1 to W
            for k = 1 to R
                if j ≥ w[i]
                    m[i,j,k] = max { m[i-1, j-w[i], k-1] + v[i] , m[i-1, j, k] }
                else
                    m[i,j,k] = m[i-1, j, k]
    return m[n,W,R]
```

subproblems = nWR

time per subproblem = O(1) update table entry

total time = O(nWR) //

2.

array $A[1..n]$ of integers (can be negative)
compute largest sum of subset of elements such that no 2 are adjacent

let $dp[i]$ store maximum sum possible in $A[1..i]$

$dp[0] = 0$
$dp[1] = \max(dp[0], A[1])$
$dp[2] = \max(dp[1], A[2])$
for $i = 3$ to n
$\quad dp[i] = \max(dp[i-1], dp[i-2] + A[i])$

return $dp[n]$

since empty subset is considered to have sum 0, $dp[0] = 0$

Base cases $dp[1]$ includes first element unless it is negative
$dp[2]$ takes larger of first 2 elements unless both negative

General case $dp[i] = \max(dp[i-1], dp[i-2] + A[i])$

If we consider the maximum sum until $i-1$, we cannot take element i as it might be adjacent $\Rightarrow dp[i-1]$

If we include element i, we can only consider maximum sum upto $i-2$
$\Rightarrow dp[i-2] + A[i]$

since we already considered $dp[0]$ empty subset to have sum 0, output will be 0 even if all elements are negative

subproblems $= n$
time per subproblem $= O(1)$
total time $= O(n)$

3.

a) n tasks, n helpers

probability helper i completes task j is $p_{i,j}$

let T be a subset of $\{1, 2, \dots n\}$ representing tasks completed

let h be # helpers involved $\{1, \dots h\}$

let $dp[T, h]$ be maximum probability of tasks in T completed by helpers $\{1 \dots h\}$

$T_{size} = h$

Base case : $h = 1$ (only helper 1 involved)

$dp[\{t\}, 1] = p_{1,t}$ for t from 1 to n

General case :

$dp[T, h] = \max \{$ (p_{h,t_1} + with helper can take take any task in subset T + max probability of other h-1 helpers assigned to $T \setminus \{t_i\}$) $dp[T \setminus \{t_1\}, h-1]$, where $T = \{t_1, t_2 \dots t_h\}$

$p_{h,t_2} + dp[T \setminus \{t_2\}, h-1]$, $O(h)$

$\vdots$

$p_{h,t_h} + dp[T \setminus \{t_h\}, h-1] \}$

for all subsets T of $\{1, 2, \dots n\}$ of size h $\binom{n}{h}$

HELPER TASK MATCHING (n, p)

for t from 1 to n

$dp[\{t\}, 1] = p_{1,t}$

for h from 2 to n

for all $T \subseteq \{1, \dots n\}$, $|T| = h$

$dp[T, h] = \max_{t_i \in T} (p_{h,t_i} + dp[T \setminus \{t_i\}, h-1])$

return $dp[T, n]$

#subproblems $= \binom{n}{1} + \binom{n}{2} + \binom{n}{3} + \dots \binom{n}{n} = 2^n - 1$

time per subproblem $= O(n)$

total time $= O(n 2^n)$

*more exact $\sum_{k=1}^{n} k \binom{n}{k} = 2^{n-1} \cdot n$

b) brute force algorithm

generate all $n!$ permutations of helpers match to tasks $O(n!)$

add all n probabilities $p_{i,j}$ and compare with current max value $O(n)$

total time = $O(n \cdot n!)$