

General Approach to Solving Optimization problems using Dynamic Programming

1. Characterize the structure of an opt. solution
2. Recursively define the value of an opt. solution
3. Compute the value of an opt. solution in a bottom up fashion
4. Construct an opt. sol. from computed information

Problem Statement

- We have 1 resource
- " " n requests labeled 1 to n
- Each request has start time s_i ,
finish time f_i , and
weight w_i

Goal: Select a subset $S \subseteq \{1..n\}$
of mutually compatible intervals
so as to Maximize $\sum_{i \in S} w_i$

Case 1 - if it is, value of the opt. sol. =
 w_i + value of the opt. sol. for
the subproblem that consists
only of compatible requests with i

Case 2 - if it isn't, value of the opt. sol. =
value of the opt. sol. without job i

Sort requests in order of non-decreasing
finish time.

$$f_1 \leq f_2 \leq \dots \leq f_n$$

Define $P(j)$ for an interval j to be the
largest index $i < j$ such that interval i & j
are disjoint.

Def. Let O_j denote the opt. solution to
the problem consisting of requests $\{1..j\}$
Let $OPT(j)$ denote the value of O_j

Solution:

Compute - opt (j)

if $j = 0$ then

return 0

else

return

A blank sheet of lined paper with a red border. The top-right corner is folded over. The page contains 12 horizontal red lines for writing.

A blank sheet of lined paper with a red border. The page contains 12 horizontal red lines for writing.

A blank sheet of lined paper with a red border. The top-right corner is folded over. The page contains 12 horizontal red lines for writing.

A blank sheet of lined paper with a red border. The page contains 12 horizontal red lines for writing.

Memorization

Store the value of compute-opt. in a globally accessible place the first time we compute it. Then simply use this precomputed value in place of all future recursive calls.

$M_compute_opt(j)$

if $j=0$ then

return 0

else if $M[j]$ is not empty then

A blank sheet of lined paper with a red border. The top-right corner is folded over. The page contains 12 horizontal red lines for writing.

A blank sheet of lined paper with a red border. The page contains 12 horizontal red lines for writing.

A blank sheet of lined paper with a red border. The top-right corner is folded over. The page contains 12 horizontal red lines for writing.

A blank sheet of lined paper with a red border. The page contains 12 horizontal red lines for writing.

Find-Solution

if $j > 0$ then

if $w_j + M[p(j)] \geq M[j-1]$ then

output j together w/ the results
of Find-Solution ($p(j)$)

else

output the results of
Find-Solution ($j-1$)

endif endif

A blank sheet of lined paper with a red border. The top-right corner is folded over. The page contains 12 horizontal red lines for writing.

A blank sheet of lined paper with a red border. The page contains 12 horizontal red lines for writing.

Videogame Problems

A blank sheet of lined paper with a red border. The top-right corner is folded over, creating a triangular flap. The paper contains 12 horizontal red lines for writing.

A blank sheet of lined paper with a red border. The paper contains 12 horizontal red lines for writing.

A blank sheet of lined paper with a red border. The top-right corner is folded over. The page contains 12 horizontal red lines for writing.

A blank sheet of lined paper with a red border. The page contains 12 horizontal red lines for writing.

A blank sheet of lined paper with a red border. The top-right corner is folded over. The page contains 12 horizontal red lines for writing.

A blank sheet of lined paper with a red border. The page contains 12 horizontal red lines for writing.

0-1 knapsack &

subset sum

Problem Statement

- A single resource
- Requests $\{1 \dots n\}$ each take time w_i to process
- Can schedule jobs at any time between $\underline{0}$ to \underline{W}

Objective: To schedule jobs such that we maximize the machine's utilization

$OPT(i, w)$ = value of the opt. solution
using a subset of the
items $\{1 \dots i\}$ with
Max. allowed weight w .

if $n \notin O$, Then $OPT(n, w) =$

if $n \in O$, Then $OPT(n, w) =$

If $w < w_i$, Then $OPT(i, w) = OPT(i-1, w)$

else, $OPT(i, w) = \text{Max}(OPT(i-1, w),$
 $w_i + OPT(i-1, w - w_i))$

Subset-sum (n, w)

array $M[0, w] = 0$ for each $w = 0$ to W

for $i = 1$ to n

for $w = 0$ to W

use recurrence formula

to compute $M[i, w]$

end for

end for

Return $M[n, w]$

A blank sheet of lined paper with a red border. The top-right corner is folded over. The page contains 12 horizontal red lines for writing.

A blank sheet of lined paper with a red border. The page contains 12 horizontal red lines for writing.

Pseudo-polynomial time

An algorithm runs in pseudo-polynomial time if its running time is a polynomial in the numeric value of the input.

Polynomial time

An algorithm runs in polynomial time if its running time is a polynomial in the length of the input (or output).

A blank sheet of lined paper with a red border. The top-right corner is folded over. The page contains 12 horizontal red lines for writing.

A blank sheet of lined paper with a red border. The page contains 12 horizontal red lines for writing.

Discussion 6

1. You are to compute the total number of ways to make a change for a given amount m . Assume that we have an unlimited supply of coins and all denominations are sorted in ascending order: $1 = d_1 < d_2 < \dots < d_n$. Formulate the solution to this problem as a dynamic programming problem.
2. Graduate students get a lot of free food at various events. Suppose you have a schedule of the next n days marked with those days when you get a free dinner, and those days on which you must acquire dinner on your own. On any given day you can buy dinner at the cafeteria for \$3. Alternatively, you can purchase one week's groceries for \$10, which will provide dinner for each day that week (that day and the six that follow). However, because you don't have a fridge, the groceries will go bad after seven days (including the day of purchase) and any leftovers must be discarded. Due to your very busy schedule, these are your only two options for dinner each night. Your goal is to eat dinner every night while minimizing the money you spend on food.
3. You are in Downtown of a city and all the streets are one-way streets. You can only go east (right) on the east-west (left-right) streets, and you can only go south (down) on the north-south (up-down) streets. This is called a Manhattan walk.
 - a) In Figure A below, how many unique ways are there to go from the intersection marked S (coordinate (0,0)) to the intersection marked E (coordinate (n,m))?
Formulate the solution to this problem as a dynamic programming problem. Please make sure that you include all the boundary conditions and clearly define your notations you use.

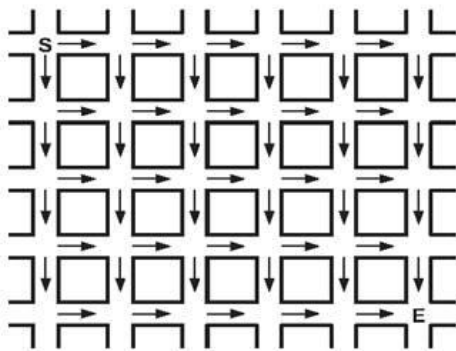


Figure A.

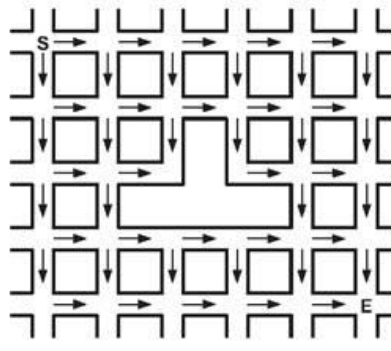


Figure B.

- b) Repeat this process with Figure B; be wary of dead ends.

A blank sheet of lined paper with a red border. The top-right corner is folded over. The page contains 12 horizontal red lines for writing.

A blank sheet of lined paper with a red border. The page contains 12 horizontal red lines for writing.

A blank sheet of lined paper with a red border. The top-right corner is folded over. The page contains 12 horizontal red lines for writing.

A blank sheet of lined paper with a red border. The page contains 12 horizontal red lines for writing.

A blank sheet of lined paper with a red border. The top-right corner is folded over. The page contains 12 horizontal red lines for writing.

A blank sheet of lined paper with a red border. The page contains 12 horizontal red lines for writing.

A blank sheet of lined paper with a red border. The top-right corner is folded over. The page contains 12 horizontal red lines for writing.

A blank sheet of lined paper with a red border. The page contains 12 horizontal red lines for writing.

A blank sheet of lined paper with a red border. The top-right corner is folded over, creating a triangular flap. The paper contains 12 horizontal red lines for writing.

A blank sheet of lined paper with a red border. The paper contains 12 horizontal red lines for writing.