Introduction to Financial Mathematics

Lecture 6: Dynamic Programming

November 23, 2024

During the lecture, we will discuss a variant of the loot sharing example and then the pots of gold example.

1 Pots of Gold

We are given an *ordered* set of numbers representing number of gold coins in that bag:

$$A = \{ x_1 , x_2 , \ldots , x_n \}.$$

There are two players, P_1, P_2 who take turns and pick one of the numbers at the edge, until all numbers are picked. For instance, at the first round P_1 can choose either x_1 or x_n.

The goal is to maximize the sum of the collected numbers. This is a zero sum game, as the sum of the number collected by either player is equal to the sum of the original set.

Example 1.1. *Suppose* $A = \{ 1 , 12 , 1 , 2 \}$.
The following scenario is within the rules of the game,

P_1 chooses left 1*, remaining set is* $A_1 = \{ 12 , 1 , 2 \}$*;*
P_2 chooses left 12*, remaining set is* $A_2 = \{ 1 , 2 \}$*;*
P_1 chooses right 2*, remaining set is* $A_3 = \{ 1 \}$*;*
P_2 chooses remaining 1*, and the game is over.*

This strategy yields 3 *coins for P_1 and it is not optimal. The strategy of P_1 to chose* 2*, then* 1 *by P_2 (either left or right), then* 12 *by P_1 and then P_2 picks the remaining* 1 *would yield* 14 *coins for P_1. Note that this is also the best for P_2 when P_1 plays optimally.*

To describe the dynamic programming approach, we introduce the following notation:

- k is the number of bags already collected, $k = 1, 2, \ldots, n$;
- i is the number of left bags collected, $i = 1, 2, \ldots, k$;
- $v_k(i)$ is the maximum coins that can be collected from this point on by the player who has the turn to pick. We assume that the other player also makes optimal decisions (if not, this player may end up with more coins);

- $w_k(i)$ is the maximum coins that can be collected from this point on by the player who will pick in the next round, if there is one. Same assumptions are made as above.

It is clear that when k is equal to the total number of bags n, then there is no remaining bag and both $v_n(i) = w_k(i) = 0$ for every i. At earlier steps, the first payer has two choices: left-most bag and right-most bag (if $k = n - 1$ both are equal and in fact there is no choice). Since i many left-bags have already been picked, the left-most bag is x_{i+1} and the right-most bag is a_{n-k+i}. In the next round, this player will have to wait and hence, the maximum possible coins available to this player would be $w_{k+1}(i + 1)$ (if this player has chosen the left bag), or $w_{k+1}(i)$ (if this player has chosen the right bag).

Hence, two choices would yield,

$$x_{i+1} + w_{k+1}(i + 1), \text{ (if left chosen)} \qquad \text{or} \qquad x_{n-k+i} + w_{k+1}(i) \text{ (if right chosen).}$$

Clearly, the optimal choice is to chose the larger one. Therefore,

$$v_k(i) = max\{\, x_{i+1} + w_{k+1}(i + 1)\,,\; x_{n-k+i} + w_{k+1}(i)\,\}, \qquad k = 0, 1, \ldots, n - 1, \;\; i = 0, 1, \ldots, k.$$

Note that the case $k = n$ is already solved and is not included in the above recursion.

To complete the description, we have to provide a recursive formula for w as well. For this goal, let $j^*_{k,i} = 1$ if the left-most bag is optimal to choose in the above step and $j^*_{k,i} = 0$ otherwise. Equivalently,

$$v_k(i) = \begin{cases} x_{i+1} + w_{k+1}(i + 1) & \text{if } \; j^*_{k,i} = 1, \\ \\ x_{n-k+i} + w_{k+1}(i), & \text{if } \; j^*_{k,i} = 0. \end{cases}$$

Moreover, the player who will pick in the next round will be in a position with $(k+1)$ bags picked and $(i + j^*_{k,i})$ of them left-most ones. Hence,

$$w_k(i) = v_{k+1}(i + j^*_{k,i}), \qquad k = 0, 1, \ldots, n - 1, \;\; i = 0, 1, \ldots, k.$$

Remark 1.2. *This example is remarkably close to the n-step Binomial model if one sees the number of left bags chosen as the number of up movements. However, in this example, we do a worst case analysis, while in the Binomial problem, hedging arguments yield an average analysis with the risk neutral measure.*

Example 1.3. *Suppose* $A = \{\, 5\,,\; 7\;,\; 12\,,\; 3\,,\; 4\,\}$.

The following scenario is within the rules of the game,

P_1 chooses left 5*, remaining set is* $A_1 = \{\, 7\;,\; 12\,,\; 3\,,\; 4\,\}$*;*
P_2 chooses left 7*, remaining set is* $A_2 = \{\, 12\,,\; 3\,,\; 4\,\}$*;*
P_1 chooses left 12*, remaining set is* $A_3 = \{\, 3\,,\; 4\,\}$*;*
P_2 chooses right 4*, remaining set is* $A_4 = \{\, 3\,\}$*;*
P_1 chooses the remaining 3 *and the game is over.*

This strategy yields 20 *coins for* P_1 *and it is better than optimal. But* P_2 *collects* 11 *coins and is not optimal. and it is better than optimal. The optimal strategy for both is this:* P_1 *to choses* 5*, then* P_2 *choses* 4*, then* 7 *by* P_1 *followed by* 12 *by* P_2 *and finally* P_1 *to choses* 3*. This would yield* 15 *coins for* P_1 *(less than the previous round) but* 16 *coins for* P_2*. Note that when a player does not play optimally the other would get more. These are called zero-sum games.*

Homework. Find the optimal strategy when the set is

a.

$$A = \{\, 15\,,\ 11\,,\ 5\,,\ 6\,,\ 8\,,\ 25\,,\ 7\ ,\ 10\,,\ 13\,,\ 4\,\}.$$

b.

$$A = \{\, 1\,,\ 2\,,\ 3\,,\ 4\,,\ 5\,,\ 6\,,\ 7\ ,\ 8\,,\ 9\,,\ 10\,\}.$$

c.

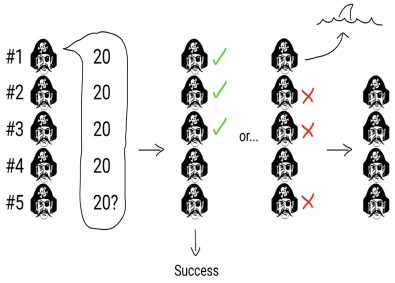
$$A = \{\, 1\,,\ 2\,,\ 3\,,\ 4\,,\ 55\,,\ 6\,,\ 7\ ,\ 8\,,\ 9\,,\ 10\,\}.$$

1.1 Python File

In the next page, the pdf file of the output of the python file is given at the end of the notes. You can use this code to run it and obtain the solutions.

2 Loot sharing

There are strictly ordered five players $p_5 > p_4 > \ldots > p_1$. They want to split 100 non-divisible units amongst them. This is done by voting on a proposal by the top ranked player. If there is a tie, or it is voted down, the proposer is removed from the list (and gets 0) and the remaining highest ranked player makes the next proposal. This is voted on and so on until a split is agreed by the remaining players. What split should the first player propose?



We make several assumptions:

- Each player is rational and there are no cliques;
- In each round, players would vote down the proposal if their optimal future share is greater or equal than the offer made in this round.

A couple of tries:

If the top player is too greedy and wants all the 100 units, then the others are not getting any and they would vote him down. So this is not optimal. Similar analysis would hold if the top player keeps 99 and gives one to one of them. Then, the other three would vote down. Suppose the top player keeps 98 and gives one to two other players. Would that be enough?

Solution.

We first generalize the problem and consider a similar problem in which there are n players $p_n > p_{n-1} > \ldots > p_1$ with $n \leq 5$. We start with $n = 1$ and work our way up to 5.

Case $n = 1$. It is clear that the sole remaining player takes it all.

Case $n = 2$. We have a pair p_2, p_1 and p_2 makes a proposal. If p_2 wants to get any non-zero amount, then p_1 would vote against. There would be a tie and p_2 is removed. Then, p_1 would get all the 100 units. So no matter what the proposal is p_1 would vote it down and gets the whole lot. The optimal sharing outcome is:

$$\text{Share of } \ p_2 = 0, \ \ \text{Share of } \ p_1 = 100.$$

Case $n = 3$. We have $p_3 > p_2 > p_1$. If the outcome of the vote is negative and p_3 is removed, then p_2 would get 0 in the next round. So p_2 would be happy to accept any non-zero offer. p_3 could calculate this and use it to his/her advantage. Then, it is optimal for p_3 to offer one unit to p_2 and take the remaining 99 units. This will be voted up by p_3 and p_2, and will consequently be accepted. So the optimal sharing outcome is as follows:

$$\text{Share of } p_3 = 99, \ \ \text{Share of } \ p_2 = 1, \ \ \text{Share of } \ p_1 = 0.$$

Case $n = 4$. We have $p_4 > p_3 > p_2 > p_1$. If the outcome in this round is negative, then in the next round p_3 will get 99, p_2 will get 1 and p_1 will get 0. The top player p_4 needs two other players to vote with him/her. The cheapest way of achieving this is to give to two players one unit more than what they would receive in the next rounds and zero to the other. It is clear that the cheapest choice is to give 1 to p_1 (she/he would accept as otherwise she/he would get nothing in the next round) and 2 to p_2. So the optimal sharing is

$$\text{Share of } p_4 = 97, \ \ \text{Share of } p_3 = 0, \ \ \text{Share of } \ p_2 = 2, \ \ \text{Share of } \ p_1 = 1.$$

Case $n = 5$. We have $p_5 > p_4 > p_3 > p_2 > p_1$. As in the above analysis, The top player p_1 needs the votes of two other players. The cheapest way of achieving is to give zero to p_4 and p_2 and one more to the others:

$$\text{Share of } p_5 = 97, \ \ \text{Share of } p_4 = 0, \ \ \text{Share of } p_3 = 1, \ \ \text{Share of } \ p_2 = 0, \ \ \text{Share of } \ p_1 = 2.$$

Note that in the above p_3 receives one more than what she/he will get in the next round and so does p_1. So the above split will be voted up by p_1, p_3, p_5.

Exercise.

a. Solve this problem when the tie-breaker goes to the proposer.
That is when there the votes in favor are equal or more than the votes against it, the proposal is accepted.

b. Solve the original problem with six players.

c. (Hard). Solve the problem with a general number of players.

3 A Knapsack Problem

This is also called the 0-1 knapsack problem, as the items cannot be divided into smaller pieces.

In this problem we are given many boxes containing valuable items. Each box weighs a certain amount and contains an item of a certain value. We have a delivery truck ti deliver them to a location. Our goal is to deliver as much as value as we can, but the truck can only carry certain weight. Then, the problem is to choose a subset of the boxes with the most value that can be carried by this truck.

3.1 An example

Each box is label by two numbers: its value and its weight. Suppose given boxes are:

$$A = \{ (20, 1) , (5, 2) , (10, 3) , (40, 8) , (15, 7) , (25, 4) \}$$

and the weight limit of the truck is 10. The first number is the value of the box and the second is its weight.

The choice $\{ (10, 3) , (15, 7) \}$ is allowed with a total value of 25. Is it optimal?

No, because the choice $\{ (20, 1) , (40, 8) \}$ is also allowed and its value is 60. Is this optimal?

3.2 General Formulation

Suppose that a non-empty set A of items is given. Each item $a \in A$ has a value $v_a > 0$ and a weight $w_a > 0$ and we write $a = (v_a, w_a)$. We are also given a weight limit of $W \geq 0$ and our goal is to choose a subset so as to maximize the total value while remaining within the weight limit. Mathematically, for any subset $B \subset A$ let

$$v(B) := \sum_{a \in B} v_a$$

be the value of the set B, and let

$$w(B) := \sum_{a \in B} w_a$$

be the weight of B. Then, the problem is to maximize $v(B)$ over all subsets satisfying the weight limit $w(B) \leq W$. Let $v(A, w)$ be the maximum possible value, i.e.,

$$v(A, W) := \max\, v(B) \qquad \text{over all subsets } B \subset A \text{ and } w(B) \leq W.$$

If there is no subset B of A with $w(B) \leq w$, then we set $v(A, w) = 0$.

3.3 Solution

An approach to this problem is to consider each subset and compute their values if they satisfy the weight limit. This method of exhaustion has high complexity as there are $2^{|A|}$ many subsets, and the number of calculations increase exponentially with the size n of the given set A. Dynamic programming is, on the hand, only linear in n. To formulate this approach, we first order the elements of A in any way:

$$A \;=\; \{\, a_1 = (v_1, w_1)\,, \ldots,\; a_n = (v_n, w_n)\,\}\,,$$

where v_i represents the value of the element a_i and w_i is its weight.

For any $k = 1, 2, \ldots, n$, and any weight limit $0 \leq w \leq W$, set

$$v_k(w) \;=\; v(\,\{a_1, \ldots,\, a_k\}\,,\, w).$$

We also set $v_0(w) \equiv\, 0$, for any w.

Theorem 3.1. *For any $k = 1, \ldots, n$ and any $w = 0, 1, \ldots, W$,*

$$v_k(w) \;=\; \begin{cases} v_{k-1}(w), & \text{if } w_k > w, \\ max\{\, v_{k-1}(w)\;;\; v_k + v_{k-1}(w - w_k))\,\}, & \text{if } w_k \leq w. \end{cases}$$

Proof. Let $A_k := \{a_1, \ldots, a_k\}$ and let $B_k(w) \subset A_k$ be an optimizer with weight limit w, i.e.,

$$v_k(w) = v(B_k(w)), \qquad \text{and} \qquad w(B_k(w) \leq w.$$

Then, for any $C \subset A_k$, $v(C) \leq v_k(w(C))$.

If $w_k > w$, a_k cannot be part of any subset satisfying the weight limit. Hence, the subsets of A_k satisfying the weight limit are exactly the subsets of A_{k-1} satisfying this weight limit. Therefore, $v_k(w) = v_{k-1}(w)$. Now, suppose that $w_k \leq w$. Set

$$C_k := \{a_k\} \cap B_{k-1}(w - w_k).$$

Clearly $C_k \subset A_k$ and as $B_{k-1}(w - w_k)$ is a maximizer with A_{k-1} and weight limit $(w - w_k)$,

$$w(C_k) = \sum_{a \in C_k} w_a = w_k + w(B_{k-1}(w)) \leq w_k + (w - w_k) = w,$$
$$v(C_k) = \sum_{a \in C_k} v_a = v_k + v(B_{k-1}(w)) = v_k + v_{k-1}(w - w_k).$$

Also, by its definition,

$$v_k(w) = \max v(B) \qquad \text{so that } B \subset A_k \text{ and } w(B) \leq w.$$

The set C_k defined above satisfies the above conditions, and hence, its value cannot be more than the maximum value $v_k(w)$. Therefore, we have argued that

$$v(C_k) = v_k + v_{k-1}(w - w_k) \leq v_k(w). \tag{3.1}$$

We continue by proving the opposite inequality. We first observe that, either $a_k \in B_k(w)$ or not. If it is the later, then $B_k(w) \subset A_{k-1}(w)$ and therefore,

$$v_k(w) \leq v_{k-1}(w), \qquad \text{when} \qquad a_k \notin B_k(w).$$

(In fact, above holds with an equality but this is not needed.) Suppose that $w_k \in B_k(w)$ and set $C_{k-1} := B_k(w) \setminus \{a_k\}$. Clearly, $C_{k-1} \subset A_{k_1}$ and

$$w(C_{k-1}) = w(B_k(w)) - w_k \leq w - w_k.$$

Hence,

$$v_k(w) = v(B_k(w)) = v_k + v_k(w(C_{k-1})) \leq v_k + v_k(w - w_k) \qquad \text{when} \qquad a_k \in B_k(w).$$

Thus,

$$v_k(w) \leq \max\{\, v_{k-1}(w)\,,\, v_k + v_{k-1}(w - w_k)\,\}.$$

Since the opposite inequality follows from (3.1), we have

$$v_k(w) \;=\; max\{\, v_{k-1}(w)\;;\; v_k + v_{k-1}(w - w_k)\,\}, \qquad \text{when } w_k \leq w.$$

□

Homework. Find the optimal strategy when the set is

a. weight limit is 9 and

$$A = \{\,(20, 5)\,,\,(5, 2)\,,\,(12, 5)\,,\,(40, 7)\,,\,(15, 5)\,\}\,.$$

b. weight limit is 6 and

$$A = \{\,(2, 1)\,,\,(3, 2)\,,\,(4, 3)\,,\,(5, 3)\,,\,(19, 5)\,,\,(5, 1)\,\}\,.$$

In [1]:
```
import numpy as np
```

Basic Functions

In [5]:
```
def bags(setA):
    # setA is the given ordered set A ={a_1,....,a_N}
    # output is the optimal values of both players
    # and the optimal action of the player who has the turn
    dim=len(setA)
    # (optimal) value when k bags left after i rights
    # first component of optimal is k-1, second is i
    value_first=np.zeros((dim,dim)) # value of the player playing
    value_second=np.zeros((dim,dim)) # value of the player waiting
    # (optimal) decision when k bags left after i rights
    # decision = 1 means take the left most one
    # decision = 0 means take the right most one
    decision=np.zeros((dim,dim))
    # single element set first , i.e., k=1
    for i in range(dim):
        value_first[0][i]=setA[i]
    for k in range(1,dim):
        for i in range(dim-k):
            left=setA[i]+value_second[k-1][i+1]
            right=setA[i+k]+value_second[k-1][i]
            value_first[k][i]= max(right , left)
            if left >= right :
                decision[k][i]=1
                value_second[k][i]=value_first[k-1][i+1]
            else:
                value_second[k][i]=value_first[k-1][i]
    return value_first , value_second , decision

# Printing

def printing_decisions(setA):
    dim=len(setA)
    vf, vs, dec = bags(setA)
    total_lefts=0
    for k in reversed(range(dim)):
        if dec[k][total_lefts] ==1:
            print("with %d bags remaining, it is optimal  to pick the left m
            print("Picked value is", setA[0])
            setA.pop(0)
            print("Remaining set is", setA)
            total_lefts +=1
        else:
            print("with %d bags remaining, it is optimal  to pick the right
            print("Picked value is", setA[-1])
            setA=setA[:-1]
            print("Remaining set is", setA)
```

Example 1

In [7]:
```
A=[ 1 , 1, 25, 1 ]
d=len(A)
F , S, D = bags(A)
print('The optimal value for the starter is',F[d-1][0])
print('The optimal value for the second player is',S[d-1][0])
print()
printing_decisions(A)
```

```
The optimal value for the starter is 26.0
The optimal value for the second player is 2.0

with 4 bags remaining, it is optimal  to pick the left most bag
Picked value is 1
Remaining set is [1, 25, 1]
with 3 bags remaining, it is optimal  to pick the left most bag
Picked value is 1
Remaining set is [25, 1]
with 2 bags remaining, it is optimal  to pick the left most bag
Picked value is 25
Remaining set is [1]
with 1 bags remaining, it is optimal  to pick the right most bag
Picked value is 1
Remaining set is []
```

In [8]:
```
A=[1 , 5,  2, 15 , 1 , 1 , 25 , 3 , 2, 8]
d=len(A)
F , S, D = bags(A)
print('The optimal value for the starter is',F[d-1][0])
print('The optimal value for the second player is',S[d-1][0])
print()
printing_decisions(A)
```

```
The optimal value for the starter is 37.0
The optimal value for the second player is 26.0

with 10 bags remaining, it is optimal  to pick the right most bag
Picked value is 8
Remaining set is [1, 5, 2, 15, 1, 1, 25, 3, 2]
with 9 bags remaining, it is optimal  to pick the right most bag
Picked value is 2
Remaining set is [1, 5, 2, 15, 1, 1, 25, 3]
with 8 bags remaining, it is optimal  to pick the left most bag
Picked value is 1
Remaining set is [5, 2, 15, 1, 1, 25, 3]
with 7 bags remaining, it is optimal  to pick the left most bag
Picked value is 5
Remaining set is [2, 15, 1, 1, 25, 3]
with 6 bags remaining, it is optimal  to pick the left most bag
Picked value is 2
Remaining set is [15, 1, 1, 25, 3]
with 5 bags remaining, it is optimal  to pick the left most bag
Picked value is 15
Remaining set is [1, 1, 25, 3]
with 4 bags remaining, it is optimal  to pick the left most bag
Picked value is 1
Remaining set is [1, 25, 3]
with 3 bags remaining, it is optimal  to pick the left most bag
Picked value is 1
Remaining set is [25, 3]
with 2 bags remaining, it is optimal  to pick the left most bag
Picked value is 25
Remaining set is [3]
with 1 bags remaining, it is optimal  to pick the right most bag
Picked value is 3
Remaining set is []
```

In [ ]: