Course Information and Introduction

Arash Rafiey

August 23, 2016

Course Information

1. **Instructor : Arash Rafiey**
   - Email : arash.rafiey@indstate.edu
   - Office : Root Hall A-127
   - Office Hours : Tuesdays, Thursdays 1:00 pm to 2:00 pm in my office (A-127)
2. **Course Webpage :**
   http://cs.indstate.edu/~arash/algo458-558.html

Course Information

Objective : Introducing concepts and problem-solving techniques that are used in the design and analysis of efficient algorithms.
How ? : By studying various algorithms and data structures.

1. Motivating examples
2. Recursive functions (how to use recursive program)
3. Greedy (graph) algorithms, BFS, DFS, Dijkstra's, Kruskal's, and Prim's
4. Simple data structures: priority queues (with heaps) and union-find
5. Divide and Conquer algorithms (solving recurrences)
6. Dynamic Programming algorithms
7. Network Flow algorithms and matching (probably)
8. NP-completeness
9. Approximation algorithms

Course Information

Textbooks :

Algorithm Design, J. Kleinberg, É. Tardos, Addison Wesley, 2006.

Introduction to Algorithms (3rd Edition), T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, MIT Press, 2009.

Grading

1. Homework assignmemts (4 assignments each 5 %)
2. Midterm (25 %)
3. Final (55 %)

Motivation by Examples

1. Recursive functions (how to use recursive program)
2. Stable Matching Problem.
3. Fair allocation problems.
4. Maze (Getting to a destination)
5. A question about link lists
6. Longest Common Substring
7. Hard problems (Maze again!)

All the subsets of a set

Given : A set S of n elements.

All the subsets of a set

Given : A set S of n elements.

Goal : Produce all the subsets of S.

All the subsets of a set

Given : A set S of n elements.

Goal : Produce all the subsets of S.

There are 2^n of them.

All the subsets of a set

Given : A set S of n elements.

Goal : Produce all the subsets of S.

There are 2^n of them.

Goal : Produce all the m-subsets of S.

Stable Matching Problem

We have a set $M = \{m_1, m_2, \ldots, m_n\}$ of men and a set $W = \{w_1, w_2, \ldots, w_n\}$ of women.

Stable Matching Problem

We have a set $M = \{m_1, m_2, \ldots, m_n\}$ of men and a set $W = \{w_1, w_2, \ldots, w_n\}$ of women.

A matching S is a set of ordered pairs from $M \times W$. Each member of M and each member of W appear at most once.

Stable Matching Problem

We have a set $M = \{m_1, m_2, \ldots, m_n\}$ of men and a set $W = \{w_1, w_2, \ldots, w_n\}$ of women.

A matching S is a set of ordered pairs from $M \times W$. Each member of M and each member of W appear at most once.

The goal is to pair the men and women (getting married).

Stable Matching Problem

We have a set $M = \{m_1, m_2, \ldots, m_n\}$ of men and a set $W = \{w_1, w_2, \ldots, w_n\}$ of women.

A matching S is a set of ordered pairs from $M \times W$. Each member of M and each member of W appear at most once.

The goal is to pair the men and women (getting married).

If we want everyone to be happy (everybody gets married) then we need a *perfect* matching S' (each member of M and each member of W appear exactly in one pair of S').

Stable Matching Problem

We have a set $M = \{m_1, m_2, \ldots, m_n\}$ of men and a set $W = \{w_1, w_2, \ldots, w_n\}$ of women.

A matching S is a set of ordered pairs from $M \times W$. Each member of M and each member of W appear at most once.

The goal is to pair the men and women (getting married).

If we want everyone to be happy (everybody gets married) then we need a *perfect* matching S' (each member of M and each member of W appear exactly in one pair of S').

Easy to do (Just pair them arbitrary)!

What if there is a preference ?

Each man $m \in M$ ranks all the women in W (we say m prefers w to w' if m ranks w higher than w').

Analogously, each women $w \in W$ ranks all the men.

What if there is a preference ?

Each man $m \in M$ ranks all the women in W (we say m prefers w to w' if m ranks w higher than w').

Analogously, each women $w \in W$ ranks all the men.

Goal : Stable Marriage (stable matching)

Suppose S is a perfect matching with the pair (m, w) and (m', w') where m prefers w' to w and w' prefers m to m' (Trouble! m, w' abandon their partners and heading off together).

In this case we say (m, w') is an *instable pair* with respect to S.

What if there is a preference ?

Each man $m \in M$ ranks all the women in W (we say m prefers w to w' if m ranks w higher than w').

Analogously, each women $w \in W$ ranks all the men.

Goal : Stable Marriage (stable matching)

Suppose S is a perfect matching with the pair (m, w) and (m', w') where m prefers w' to w and w' prefers m to m' (Trouble! m, w' abandon their partners and heading off together).

In this case we say (m, w') is an *instable pair* with respect to S.

Definition

Matching S is stable if it is perfect and it has no instable pair.

What if there is a preference ?

Each man $m \in M$ ranks all the women in W (we say m prefers w to w' if m ranks w higher than w').

Analogously, each women $w \in W$ ranks all the men.

Goal : Stable Marriage (stable matching)

Suppose S is a perfect matching with the pair (m, w) and (m', w') where m prefers w' to w and w' prefers m to m' (Trouble! m, w' abandon their partners and heading off together).

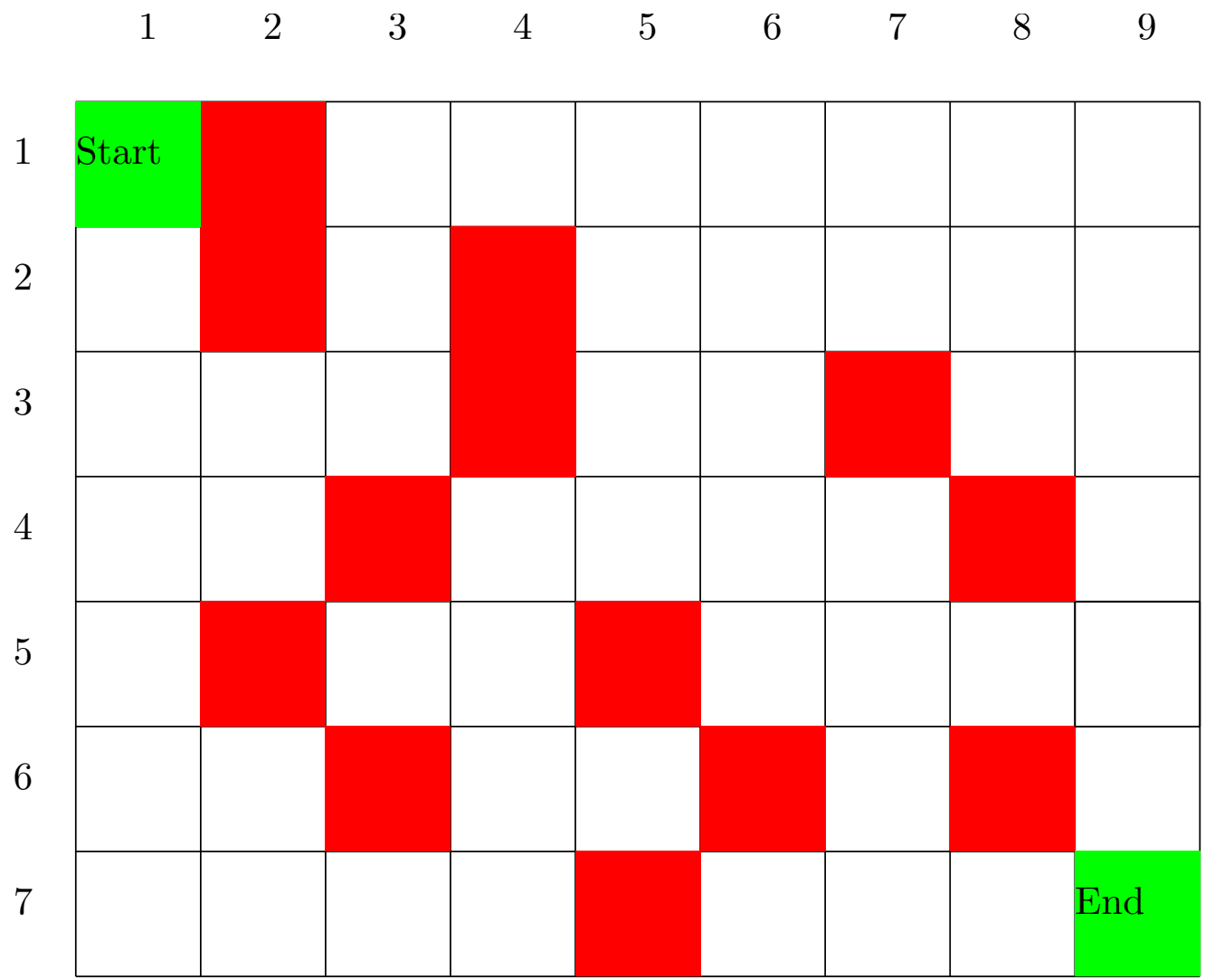
In this case we say (m, w') is an *instable pair* with respect to S.

Definition

Matching S is stable if it is perfect and it has no instable pair.

1. Does there exists a stable matching for every set of preferences ?
2. Given a set of preferences, can we (efficiently) construct a stable matching ?

Maze (Getting to a destination)



Maze (Getting to a destination)

	1	2	3	4	5	6	7	8	9
1	Start 0		6	7	8				
2	1		5		9				
3	2	3	4		10				
4	3	4		12					
5	4		12	11					
6	5	6		10	11				
7	6	7	8	9					End

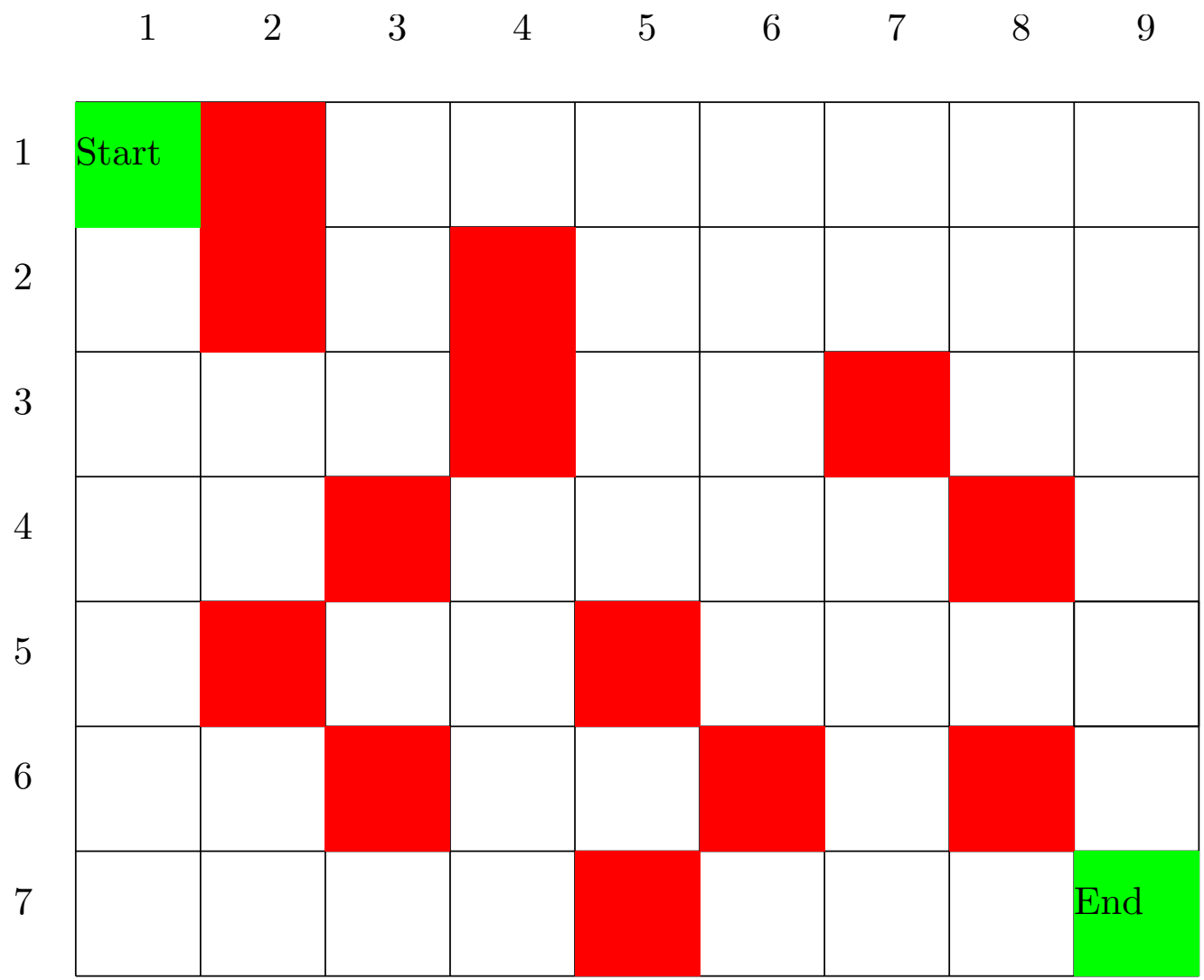
Maze (Getting to a destination)

	1	2	3	4	5	6	7	8	9
1	Start 0		6	7	8				
2	1		5		9	10			
3	2	3	4		10				
4	3	4		12	11				
5	4		12	11					
6	5	6		10	11				
7	6	7	8	9					End

Maze (Getting to a destination)

	1	2	3	4	5	6	7	8	9
1	Start 0		6	7	8	9	10	11	12
2	1		5		9	10	11	12	13
3	2	3	4		10	11		13	14
4	3	4		12	11	12	13		15
5	4		12	11		13	14	15	16
6	5	6		10	11		15		17
7	6	7	8	9		17	16	17	End

Maze Again (How do we find a longest path from start to end?)



Link List Question (Reminder)

Decide whether a link-list has a cycle ?

Link List Question (Reminder)

Decide whether a link-list has a cycle ?

Set two pointers p, q at the beginning of the link list. Move p twice faster than q. If p and q see each other then there is a loop in the link-list.

Dynamic Programming

Two examples !

- Computing $\binom{n}{m}$
- Computing longest common substring

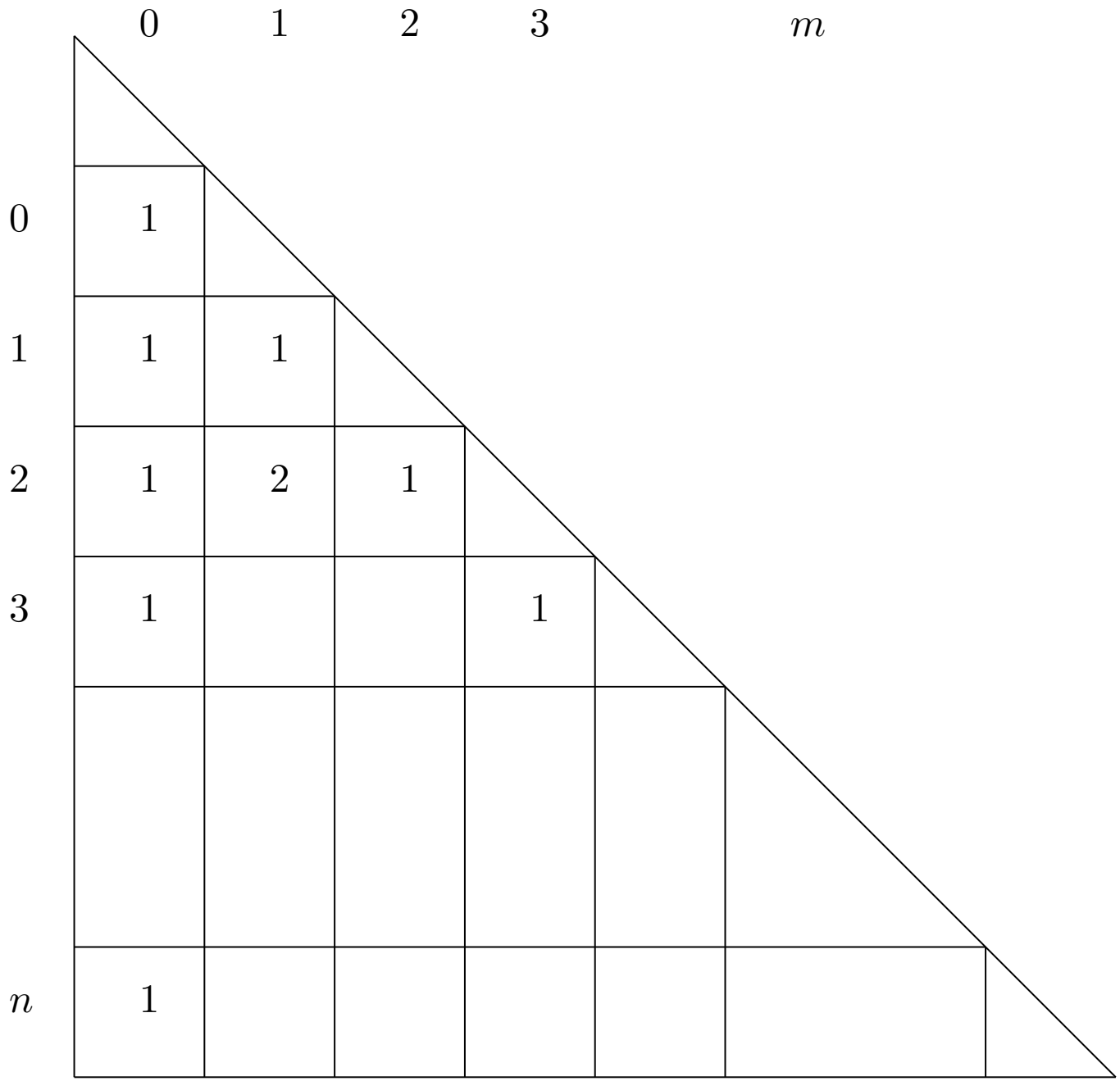
$$\binom{n}{m} = \frac{n!}{m!(n-m)!}$$

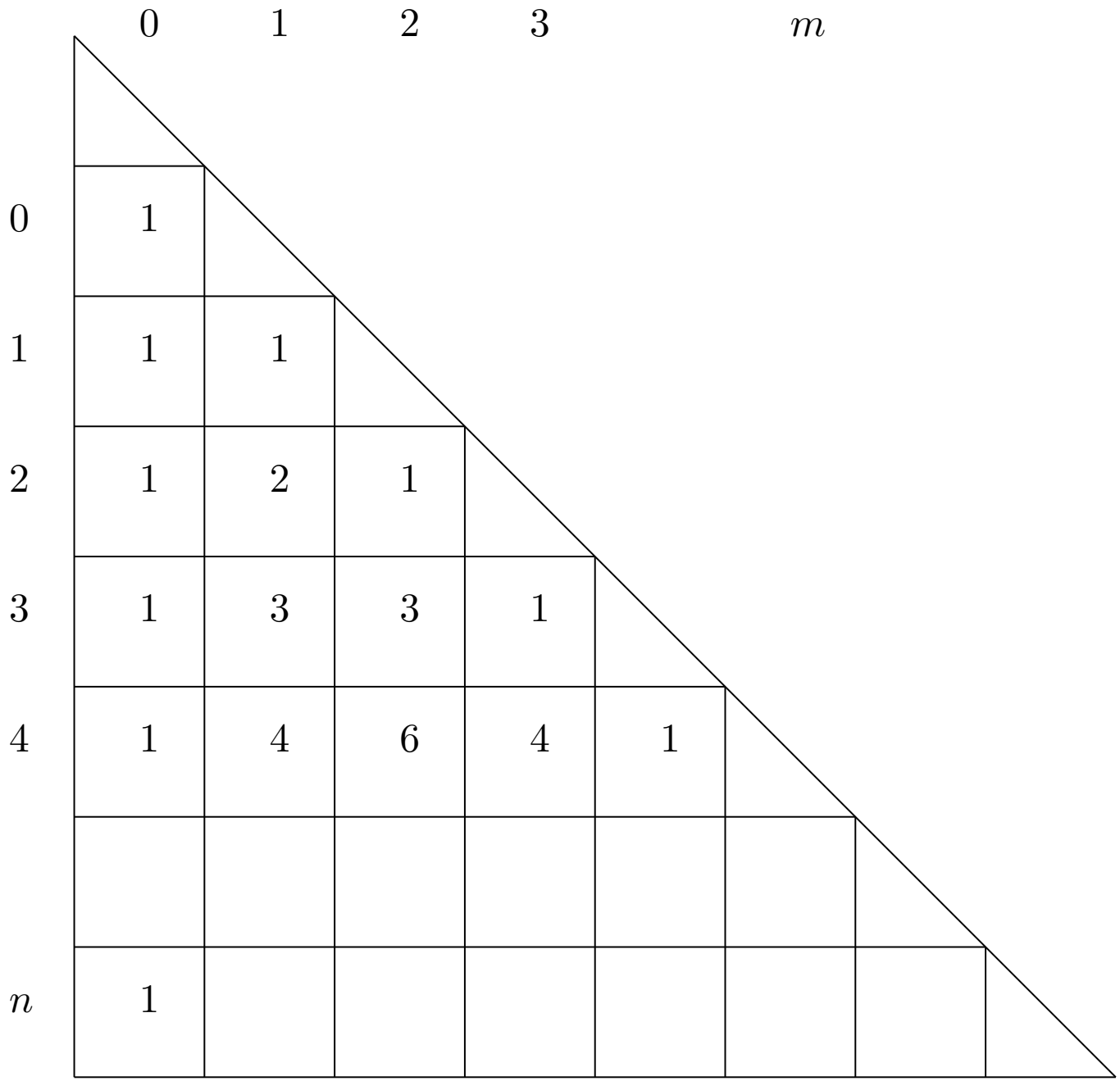
Suppose we want to avoid multiplication (costly!). We just want to use summation.

$$\binom{n}{m} = \frac{n!}{m!(n-m)!}$$

Suppose we want to avoid multiplication (costly!). We just want to use summation.

$\binom{n}{m} = \binom{n-1}{m} + \binom{n-1}{m-1}$.
$\binom{n}{n} = 1$, $\binom{n}{0} = 1$ and $\binom{n}{1} = n$.


0 1 2 3 m
0 1
1 1 1
2 1 2 1
3 1 1
n 1


0 1 2 3 m
0 1
1 1 1
2 1 2 1
3 1 3 3 1
4 1 4 6 4 1
n 1