DISCRETE STRUCTURES II 2804

DR. J.-R. SACK

CHANCELLOR’S PROFESSOR

CARLETON UNIVERSITY

SACK@SCS.CARLETON.CA

PROFESSOR INFORMATION

Professor	Office	Email	Office Hours
Dr. J.-R. Sack	HP 5350 (currently not)	sack@scs.carleton.ca	Fridays 9:00-10:00 Via Zoom (https://carleton-ca.zoom.us/j/98486347428) or ask for an appointment via email

2804 DESCRIPTION

cs.colostate.edu

A second course that is designed to give students a basic understanding of Discrete Mathematics and its role in Computer Science. Computers handle discrete data rather than continuous data. The course presents an overview of some of the major theoretical concepts needed to analyze this type of data.

PREREQUISITE:

COMP 1805 or MATH 1800, with a minimum grade of C-.
Counting, sequences and sums, discrete probability, basic statistics, recurrence relations, randomized algorithms. Material is illustrated through examples from computing.

Make sure that you really know the material from 1805 or equivalent. This will make it this course a lot more enjoyable.

The course material is crucial for 3804.

CLASS SCHEDULE

Classroom	**Via Zoom**
Class Times	Mondays and Wednesdays 10:05-11:25
Course Website	The material will be available on CULearn **Material**: Course notes and videos recording the lectures. I do not mind if you watch videos from other instructors. Different styles very similar material. E.g., http://cglab.ca/~michiel/2804.html or http://www.cglab.ca/~morin/teaching/2804/

TEACHING ASSISTANTS

TA	Office Hours via Zoom Link see CULearn (may change)
Mehrnoosh Javarsineh	Fridays 10:00-13:00 https://carleton-ca.zoom.us/j/91836593000
Denis Po	Tuesdays 16:00 - 18:00 https://carleton-ca.zoom.us/j/95033509658
Jason Dunn	Wednesday 10:30 – 12:30 https://carleton-ca.zoom.us/j/3965817406
Karim Hurani	Friday 18:00 – 20:00 https://carleton-ca.zoom.us/j/91836593000
Francois-Xavier Chaplain Corriveau	Mondays 13:00– 17:00 https://carleton-ca.zoom.us/j/2201080217
Abdulaah Emin	Wednesdays 16:00 – 17:00 & https://carleton-ca.zoom.us/j/7804790046 Thursdays 16:00– 17:00 https://carleton-ca.zoom.us/j/7804790046

RECOGNIZED STUDY GROUPS (RSGS)

(SLIDE PROVIDED BY SCIENCE STUDENT SUCCESS CENTRE)

- **Become an RSG Leader or Participant**
- **Study for class with your peers on a set schedule and earn CCR credit**
- **RSG Leader deadline: Friday, February 12th**
- **RSG Participant deadline: Friday, February 19th**
- **Find out more at sssc.carleton.ca/recognized-study-groups**

RECOGNIZED STUDY GROUPS

Science Student Success Centre
CARLETON UNIVERSITY

TOPICS

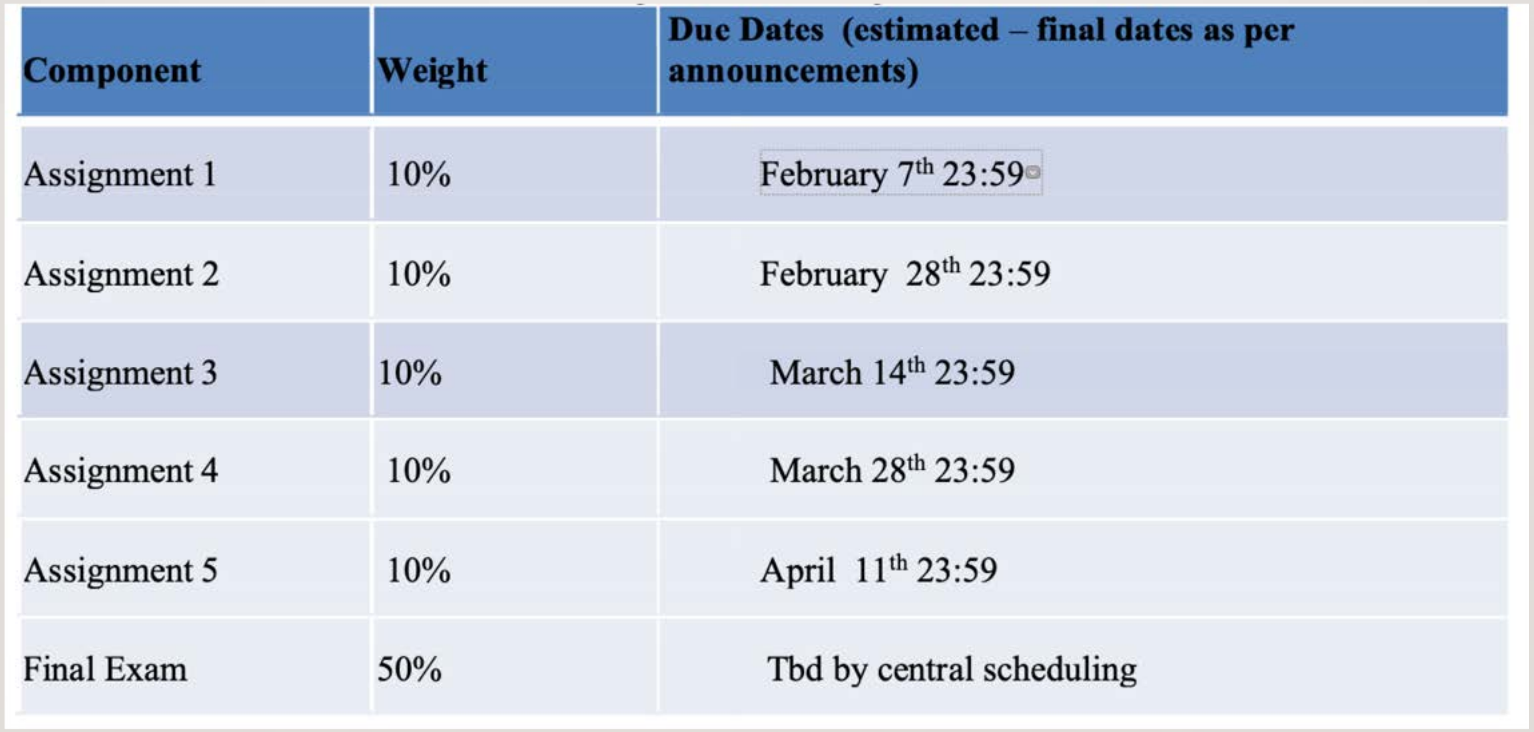
TOPICS COVERED

Counting, sequences and sums, discrete probability, basic statistics, recurrence relations, randomized algorithms. Material is illustrated through examples from computing.

TEXTBOOK

- Discrete Structures for Computer Science: Counting, Recursion, and Probability by Michiel Smid
- Additional material: Discrete Mathematics and its Applications, by Rosen. This book contains a large number of exercises as well as solutions. We used to use this book for COMP 1805, so you may still have an old copy, or you may borrow a copy from a friend.

EVALUATION



Component	Weight	Due Dates (estimated – final dates as per announcements)
Assignment 1	10%	February 7th 23:59
Assignment 2	10%	February 28th 23:59
Assignment 3	10%	March 14th 23:59
Assignment 4	10%	March 28th 23:59
Assignment 5	10%	April 11th 23:59
Final Exam	50%	Tbd by central scheduling

There may be an additional in-class midterm which has 0% weight.

ASSIGNMENTS

- Assignments will be submitted online in pdf via CU-learn (details to be provided). Typed assignments are preferred. Figures may be drawn and scanned in. It is your responsibility to ensure that the quality of the pdf is good so that the TAs can read them easily. Scanners are accessible in many locations. Pictures taken with mobile devices must be of top quality.

- Late assignments will not be accepted. Late is after the deadline specified.

COLLABORATION POLICY

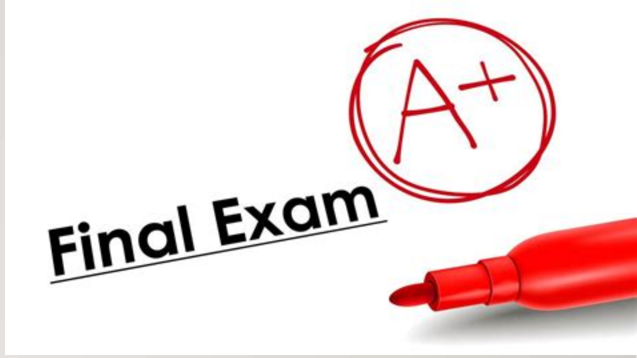
- on class-level; see also university rules below
- Students are encouraged to collaborate on assignments, but at the level of discussion only. When writing down the solutions, students must do so in their own words.
- Senate policy states that "to ensure fairness and equity in assessment of term work, students shall not co-operate or collaborate in the completion of an academic assignment, in whole or in part, when the instructor has indicated that the assignment is to be completed on an individual basis".

geekstechnicalsupport.wordpress.com

TECHNICAL PROBLEMS

- do not exempt you from this requirement, so if you wait until the last minute and then have issues with your connection, you will still receive a mark of zero.
- Consequently, you are advised to:
  - periodically upload you progress (e.g. upload your progress at least daily)
  - attempt to submit your final submission at least one hour in advance of the due date and time

FINAL EXAM



- The final exam will be scheduled by the University.
- There are many sample questions available from various sources e.g., other instructor at Carleton.
- The final will be joint for the two sections.

UNDERGRADUATE ACADEMIC ADVISOR

The Undergraduate Advisor for the School of Computer Science is available in Room 5302C HP; by telephone at 520-2600, ext. 4364; or by email at undergraduate_advisor@scs.carleton.ca. The undergraduate advisor can assist with information about prerequisites and preclusions, course substitutions/equivalencies, understanding your academic audit and the remaining requirements for graduation. The undergraduate advisor will also refer students to appropriate resources such as the Science Student Success Centre, Learning Support Services and Writing Tutorial Services.

SCS COMPUTER LABORATORY (MAY BE CLOSED DUE TO CORONA)

SCS students can access one of the designated labs for your course. The lab schedule can be found at: https://carleton.ca/scs/tech-support/computer- laboratories/. All SCS computer lab and technical support information can be found at: https://carleton.ca/scs/technical-support/. Technical support is available in room HP5161 Monday to Friday from 9:00 until 17:00 or by emailing support@scs.carleton.ca.

UNIVERSITY POLICIES

For information about Carleton's academic year, including registration and withdrawal dates, see Carleton's Academic Calendar.

UNIVERSITY POLICIES CONT'D

- Pregnancy Obligation.
  Please contact your instructor with any requests for academic accommodation during the first two weeks of class, or as soon as possible after the need for accommodation is known to exist. For more details, visit Equity Services.
- Religious Obligation.
- Please contact your instructor with any requests for academic accommodation during the first two weeks of class, or as soon as possible after the need for accommodation is known to exist. For more details, visit Equity Services.

UNIVERSITY POLICIES CONT'D

Academic Accommodations

for Students with Disabilities If you have a documented disability requiring academic accommodations in this course, please contact the Paul Menton Centre for Students with Disabilities (PMC) at 613-520-6608 or pmc@carleton.ca for a formal evaluation or contact your PMC coordinator to send your instructor your Letter of Accommodation at the beginning of the term. You must also contact the PMC no later than two weeks before the first in-class scheduled test or exam requiring accommodation (if applicable). After requesting accommodation from PMC, meet with your instructor as soon as possible to ensure accommodation arrangements are made. For more details, visit the Paul Menton Centre website.

Survivors of Sexual Violence.

As a community, Carleton University is committed to maintaining a positive learning, working and living environment where sexual violence will not be tolerated, and survivors are supported through academic accommodations as per Carleton's Sexual Violence Policy. For more information about the services available at the university and to obtain information about sexual violence and/or support, visit: carleton.ca/sexual-violence- support

UNIVERSITY POLICIES CONT'D

Accommodation for Student Activities.

- Carleton University recognizes the substantial benefits, both to the individual student and for the university, that result from a student participating in activities beyond the classroom experience. Reasonable accommodation must be provided to students who compete or perform at the national or international level. Please contact your instructor with any requests for academic accommodation during the first two weeks of class, or as soon as possible after the need for accommodation is known to exist. For more details, see the policy.
- **Student Academic Integrity Policy.**
- Every student should be familiar with the Carleton University student academic integrity policy. A student found in violation of academic integrity standards may be awarded penalties which range from a reprimand to receiving a grade of F in the course or even being expelled from the program or University. Examples of punishable offences include: plagiarism and unauthorized co-operation or collaboration. Information on this policy may be found here.

UNIVERSITY POLICIES CONT'D

Plagiarism.

As defined by Senate, "plagiarism is presenting, whether intentional or not, the ideas, expression of ideas or work of others as one's own". Such reported offences will be reviewed by the office of the Dean of Science.

Medical Certificate.

The following is a link to the official medical certificate accepted by Carleton University for the deferral of final examinations or assignments in undergraduate courses. To access the form, please go to http://www.carleton.ca/registrar/forms

SUMMARY OF LECTURES

In these notes, we will not write down all the details presented in the lectures as you have the (free) textbook and the recorded lectures.

We will attempt to get you engaged in the lectures to provide a deeper learning process and help to get prepared for subsequent classes.

2804 is a very useful course, in general, and of critical importance for 3804!

CHAPTER I

Introduction

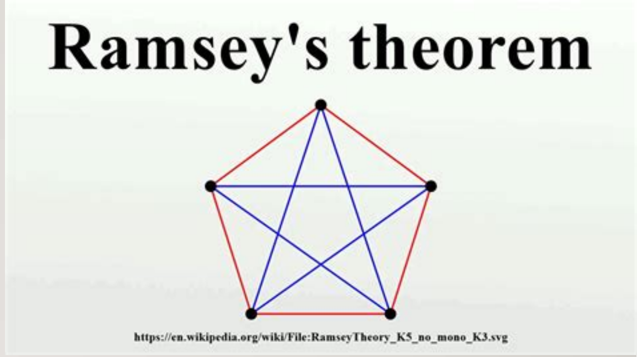
- Ramsey Theory
- Spener’s Theorem
- Quick-Sort

Introduction

INTRODUCTION

- This chapter is to give you a glance at some of the problems we will study later.
- There are some interesting concepts and recollection of proof techniques that you will see as well.

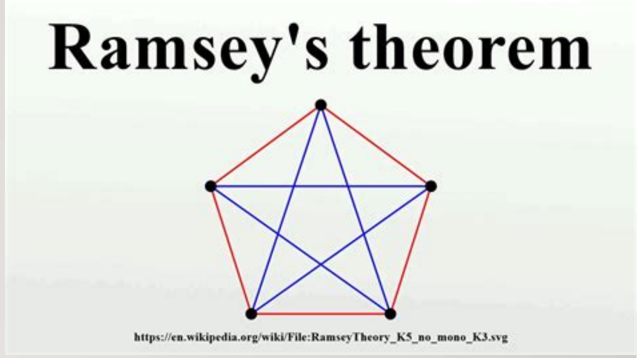
RAMSEY THEOREM


Ramsey's theorem
https://en.wikipedia.org/wiki/File:RamseyTheory_K5_no_mono_K3.svg

- Ramsey theorem are problems of this form:
  - How many elements of a given type must there be so that we can guarantee certain properties. Typically Ramsey theorem is used also for larger number of elements.

  - Elements – people
  - Property – being friends or strangers
  - So, the question could be in a large group of people is there a large clique of friends or a large groups of strangers.

RAMSEY – CONT'D


Ramsey's theorem
https://en.wikipedia.org/wiki/File:RamseyTheory_K5_no_mono_K3.svg

- So, what is a friend/stranger?
- What is a clique?
- What is large?

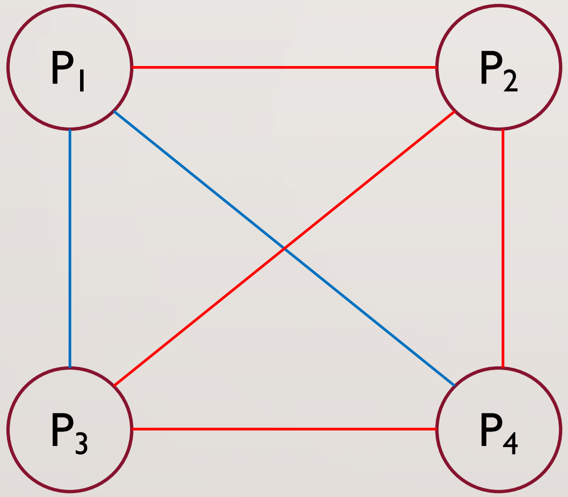
- **Definition** (somewhat arbitrary) we say that two people are friends if they are facebook friends; otherwise, we call hey are strangers.
- The importance is here only that the relations of being friends and strangers are mutually exclusive, symmetric and one of these holds for every pair of persons.

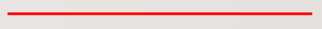
GRAPH REPRESENTATION

- Both relations could be represented in a single complete and undirected graph.

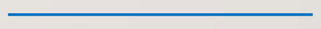
  (recall 1805 and see lecture)

4 persons $P_1, \ldots, P_4$





denotes: friends



denotes: strangers

So here: e.g., P_2 and P_4 are friends and P_1 and P_3 are strangers.

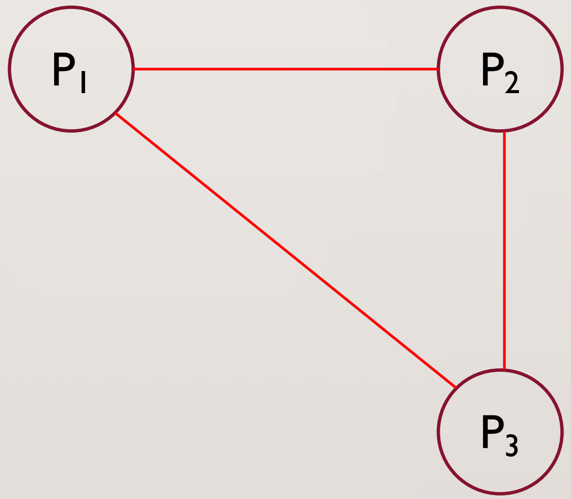
CLIQUE

- What is then a clique?
  - Colloquially it is a group of mutual friends
  - In graph terms it is a subgraph that is complete
  - Here is it a subgraph that is complete and monochromatic (all edges same colour).
  - A k-clique is a clique of k elements
  - Examples: see class
  - 3-clique is

THEOREM 1.1.1

Theorem 1.1.1 In any group of six people, there are

- there is a 3-clique of friends or
- a 3-clique of strangers.



Ex: 3-clique of friends

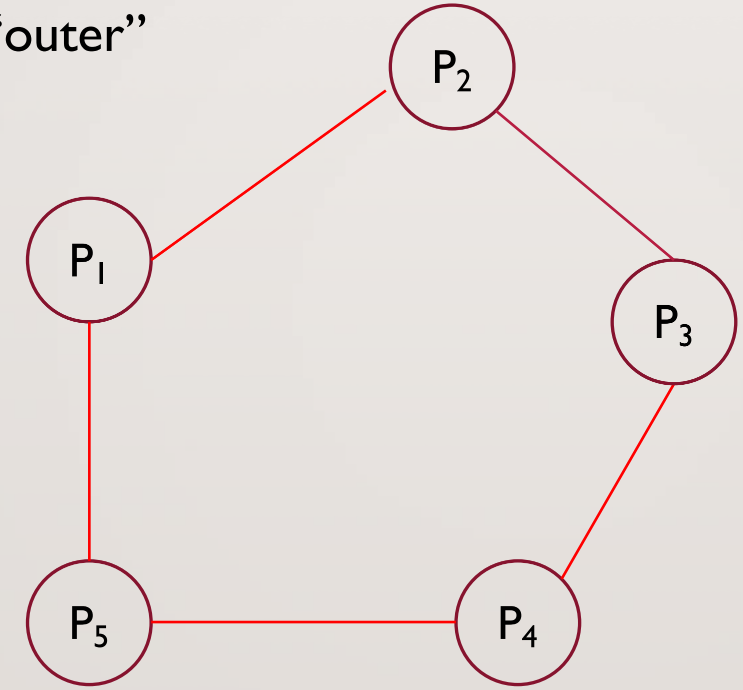
BEFORE WE PROVE THIS

- Is the theorem true for smaller groups of 3, 4, or 5 elements?
- Answers see class, esp. the case of 5 is interesting to understand the Theroem.

EXAMPLE GROUP OF 5 PERSONS

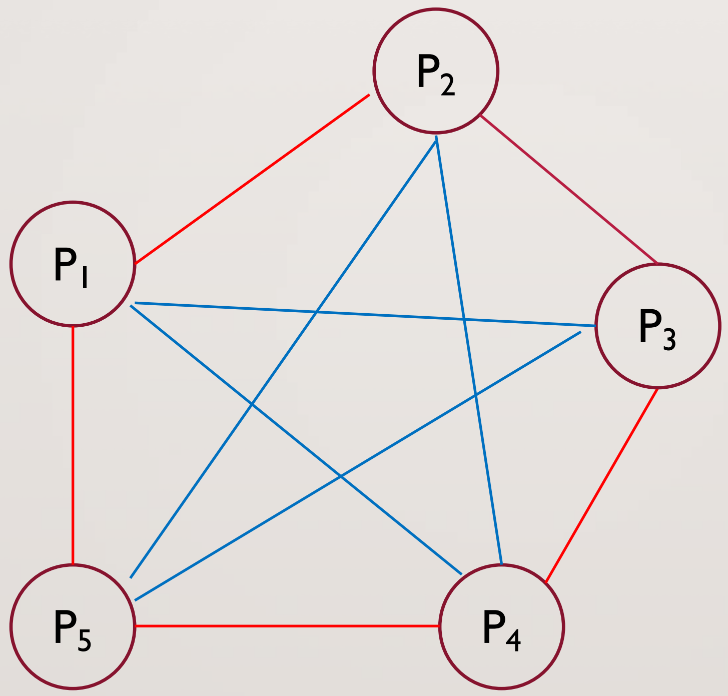
Let us colour the "outer"

edges all red.



We try to make an analogous theorem not Hold for the case of 5.
So, we can neither have have red nor blue triangles (representing cliques).

EXAMPLE GROUP OF 5 PERSONS



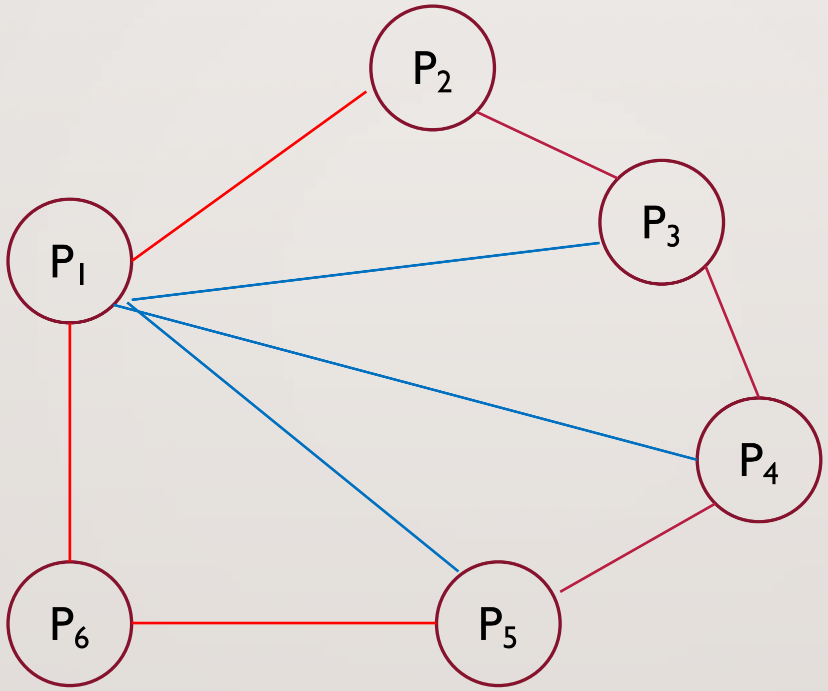
To avoid getting red triangles, all “inner” edges must be blue.

Why?

Is there a monochromatic 3-clique?

So, the theorem does not hold for the case of 5.

WHAT IS THEN DIFFERENT FOR 6?



We start with the outer edges red.
Let us look at P_1 and (forced) connect to all other vertices with blue edges.

How should we colour the edge connecting P_1 and P_1 ?
Red? – then ….
Blue? - then ….

So, what is the difference to 5?

BACK TO THEOREM 1.1.1

Theorem 1.1.1 In any group of six people, there are

- there is a 3-clique of friends or
- a 3-clique of strangers.

How to prove this now? Let us get a help.

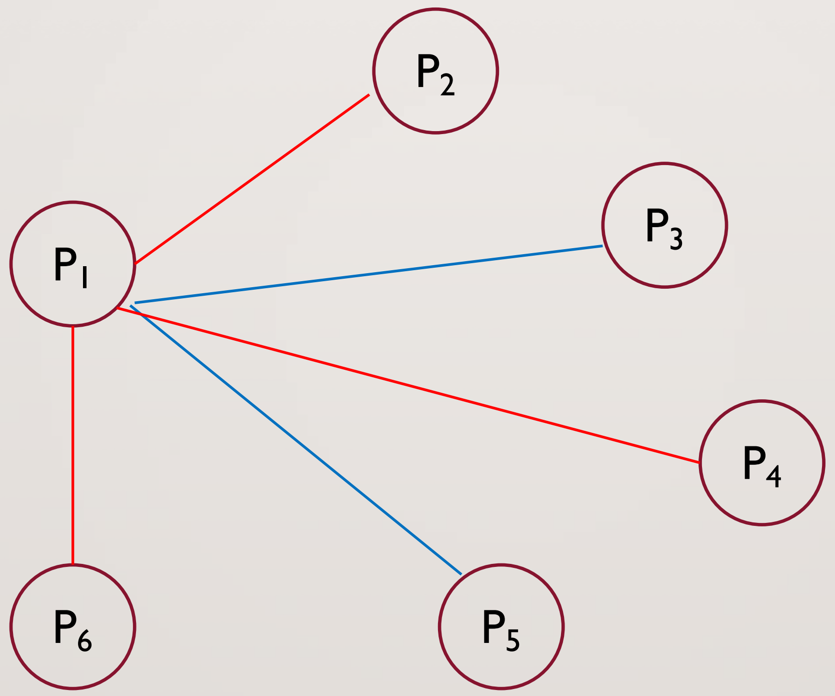
HELP USING THIS THEOREM

Theorem 1.1.2: any complete graph of 6-vertices, with edges coloured red or a blue, contains a red or a blue triangle.

Proof. See class or textbook

The key is the following:

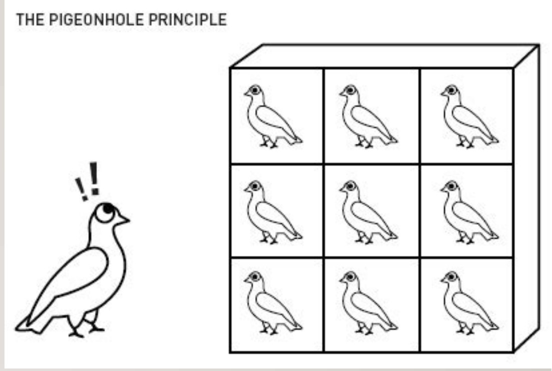
KEY IDEA



P_1 is connected either via at least 3 red or at least 3 blue edges.
Why?

See class
2 proof techniques
by contradiction
via Pigeon hole principle

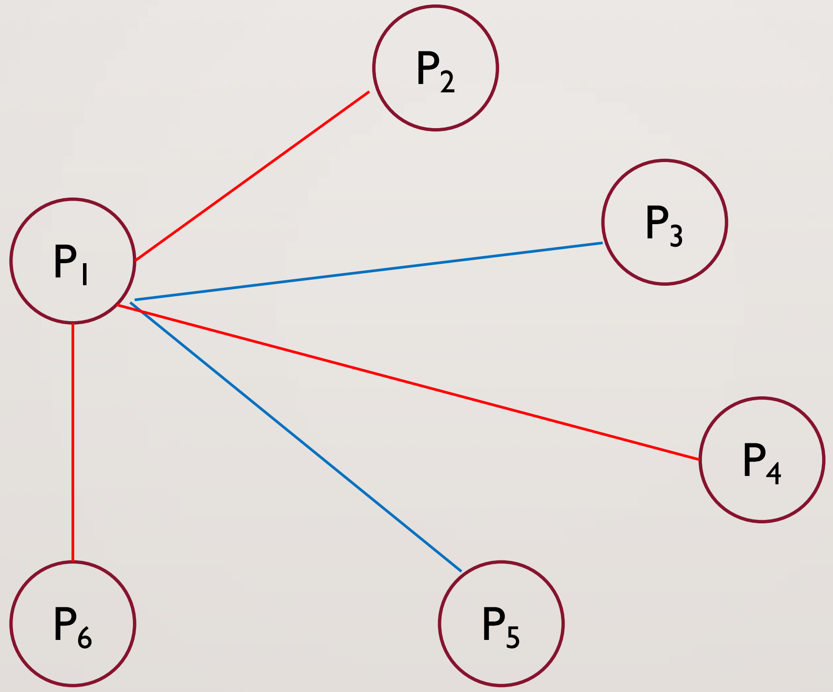
PIGEONHOLE PRINCIPLE



If n pigeonholes are occupied by n+1 or more pigeons, then at least one pigeonhole is occupied by greater than one pigeon.

We will prove many interesting results using this seemingly trivial principle.

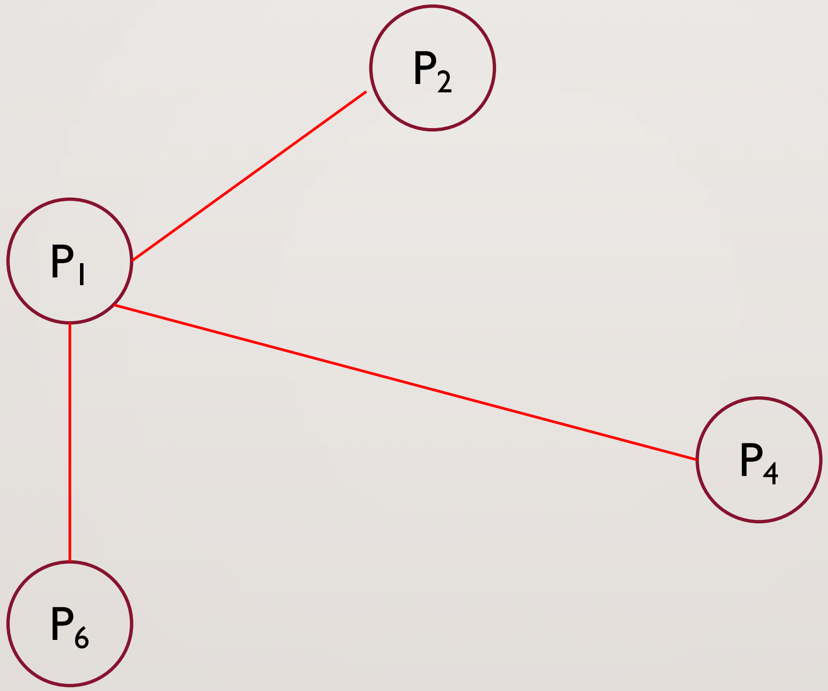
KEY IDEA



Here: P_1 is connected via 3 edges.

Forget the blue edges and incident vertices.

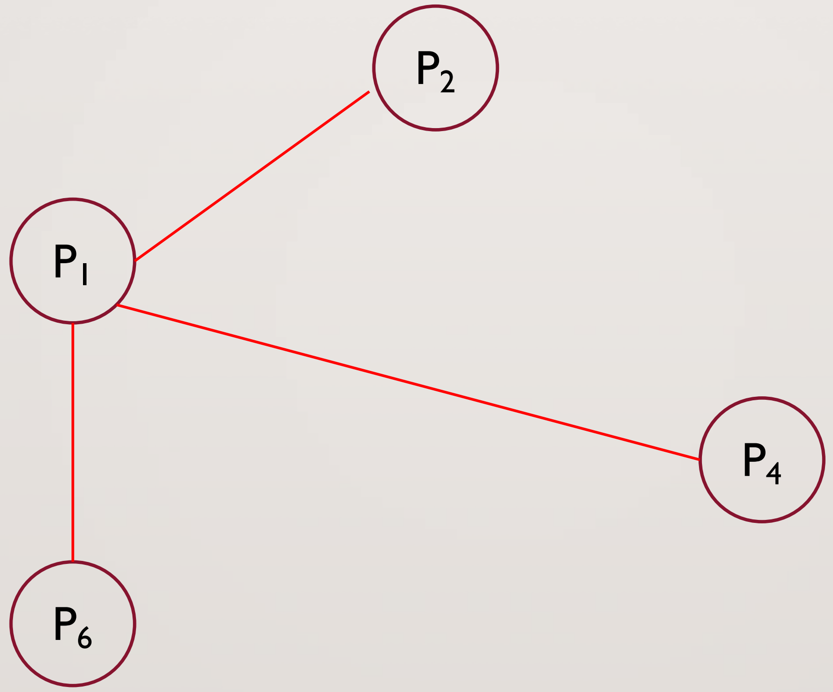
KEY IDEA



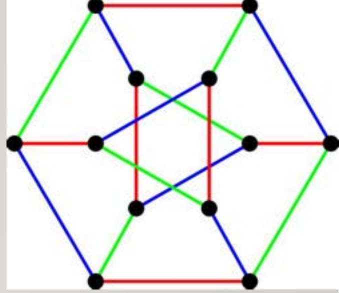
Here: P_1 is connected via 3 edges.

Forget the blue edges.

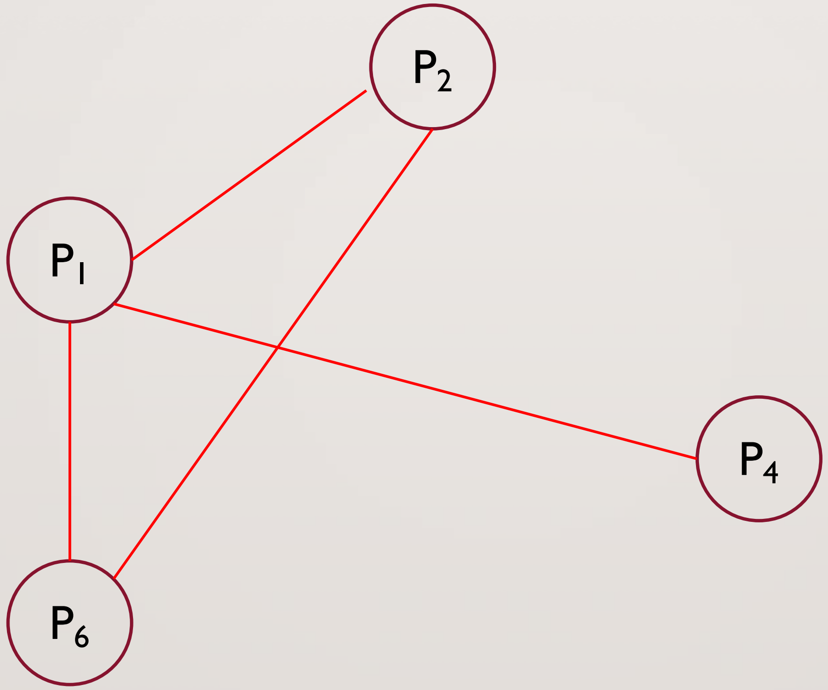
KEY IDEA



Now how to colour the missing edges when inserting the remaining edges of this (complete) subgraph?



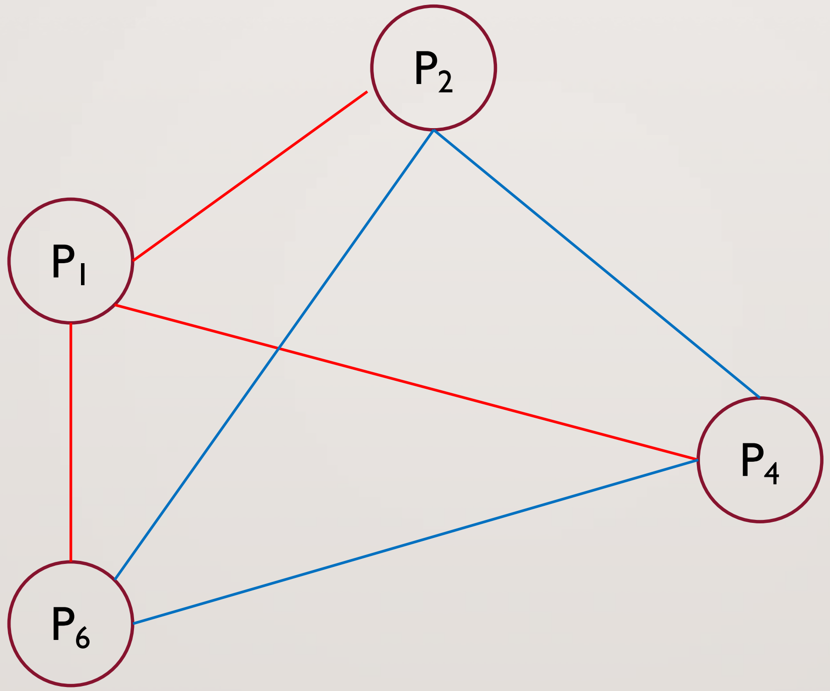
COLOUR REMAINING EDGES



If any of the remaining edges is red, then we have a monochromatic triangle.

Example: say P_2 to P_6.

COLOUR REMAINING EDGES



So, if we want to make the theorem ont hold we are force colour them all blue.

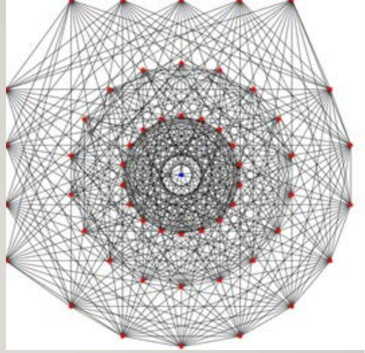
We then, we have creatad a blue triangle.

Thus, we either get a red or a blue triangle.

End of proof.

PROOF THEOREM 1.1.1

Proof. Now follows from Theorem 1.1.2.



HOW ABOUT LARGE SIZED GRAPHS?

We will show (later) that the following theorem holds which is an example of the Ramsey type problems that are useful.

Theorem 1.1.3 Let $k \geqq 3$ and $n \geqq 3$ be integers with $2^{\lfloor k/2 \rfloor} \geqq n$. There exists a complete graph with n vertices, in which each edge is either red or blue, such that this graph does not contain a monochromatic k-clique.

The proof will be via probability theory.

Take e.g., $k=20$ and $n-1024$. If you are in a group of 1024 persons then with high probability this group does neither contain a clique of 20 friends nor of 20 strangers. So, at least 2 friends and 2 strangers in that group.

SPERNER'S THEOREM

Let $n \geqq 1$ be an integer and let S be a set with n elements. Let $S_1, S_2, \ldots, S_m$ be a sequence of m subsets of S, such that for all i and j, with $i \neq j$,

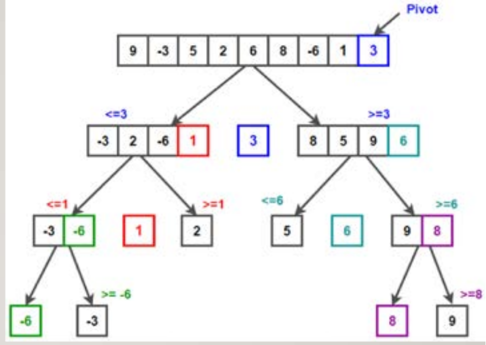
$$S_{i}, \not\subseteq S_j \text{ and } S_{i}, \not\subseteq S_j \quad \text{then}$$

$$m \leqq \binom{n}{\lfloor n/2 \rfloor}.$$

Examples: see class

We will prove this theorem using elementary counting and probability theorem.

1.3 QUICK-SORT



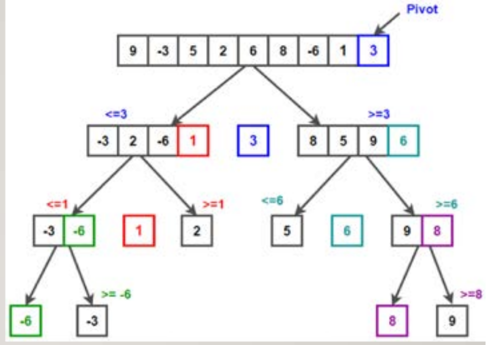
One of the fastest (clock-time) sorting algorithms

May be 2-3 times faster than competitors (depending on implementation)

Tony Hoare (1959) published in 1961

Heapsort was invented by J. W. J. Williams in 1964

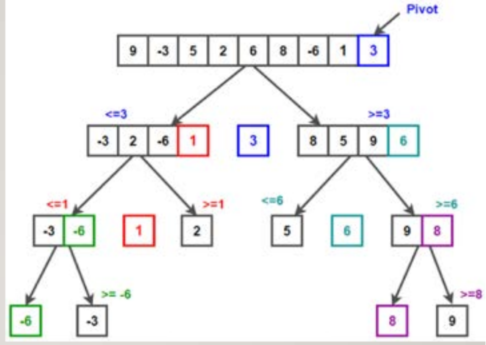
QUICKSORT



Input: a set S of n numbers stored in an array

Output: the array with the n input numbers in sorted order (say increasing)

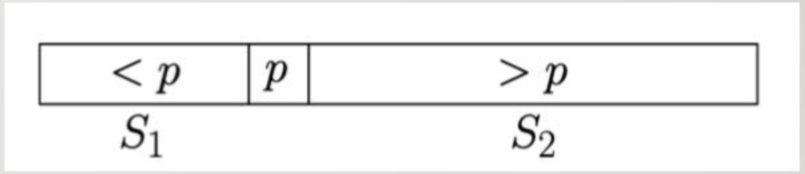
QuickSort is an in-place algorithm (or can be): so no additional array required, just constant additional storage location



QUICKSORT

Algorithm

```
if n= 0 or n =  1 return (nothing to do)
if n ≧ 2
        pick a number, call it the pivot, p
        partition S into three subsequences
        $S_1$ : all elements less than p
            all elements that are equal to p  (there could be several occurrences of p)
        $S_2$ : all elements greater than p
        recursively call QuickSort on both $S_1$ and $S_2$
```



PARTITION S INTO THREE SUBSEQUENCES

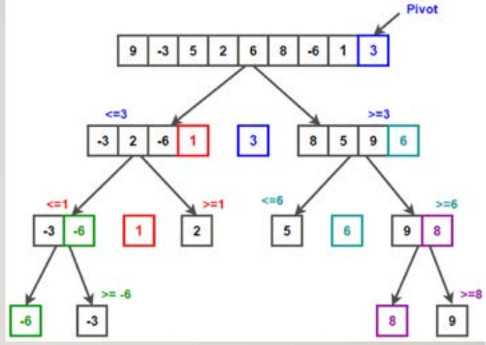
How is partition done?

we examine each element of the array expect for the pivot and ask if it <, =, > than the pivot p.

What is its number of operations?

n-1

see class or book



QUICKSORT

Does QuickSort work?

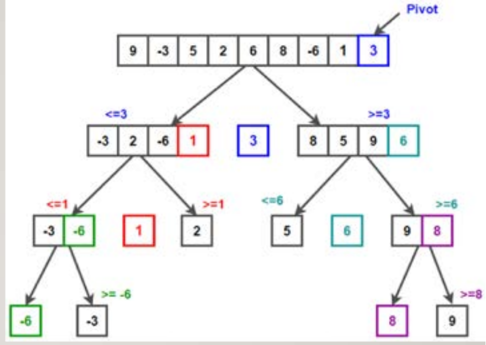
the proof of this is easy and omited here

What is the run-time?

that depends heavily on the choice of pivot as we will see

Can we do better in term worst-case number of operations carried out?

yes – HeapSort and other algorithms are better in that respect



QUICKSORT

But in clock-time it is very fast.
Here is why!

First though what are bad instances for QuickSort?

That depends on the pivot.

To make the analysis simpler to state we assume that all elements are distinct. We can easily remove that restriction.

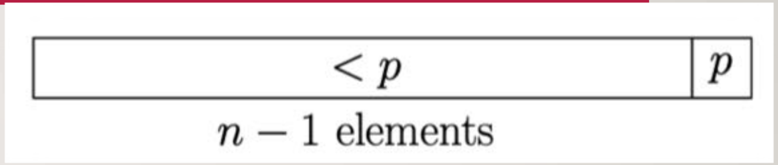
CHOICE OF PIVOT IS CRUCIAL FOR QUICKSORT

1. Assume that the pivot is the smallest number in the input array.

   Then, S_1 is of size 0.

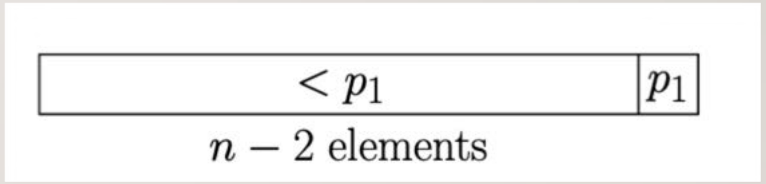
WORST-CASE OF QUICKSORT

In this case, S_2 contains all elements but p. So, its size is n-1.



We recurse on S_2, say with pivot p_1, and, if we are again unlucky, the analogous situation arises.

This now took n-2 comparisons.



…

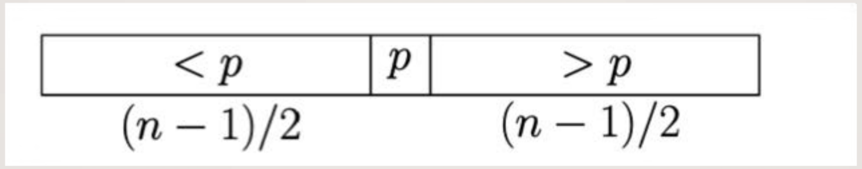
WORST-CASE CONT'D

- so, when we are done we executed
    - n-1 + n-2 + n-3 + … + 1 comparisons.
    - This sum is n*(n-1)/2 quadratic in n (see Section 2.3 for the notation of $\Theta(n^2)$)
    - Here is means that the number of comparisons taken by QuickSort is quadratic in the input size n.

    - Note: I count comparisons the textbook uses the vague notion of steps. Not very relevant for the discussion except there is small difference in one of the terms of the sum.

WHAT IS A GOOD CASE FOR QUICKSORT?

- If the pivot happens to be splitting the array nicely into two roughly equal parts.



- In that case, see Section 4.6, QuickSort will indeed be quick and execute using a number of comparisons that is proportional to n $\log_2$ n. (Also, the data movements will be not be more.) (see Section 2.3 for the notation O(n $\log_2$ n)).

EXAMPLE

- Let us say that n = 1, 000,000 (for big data this is nothing special)
- Then, n(n-1)/2 = 1,000,000*999999/2= 499,999,500.000   499 billion
- A fast computer execute say 1 billion comparisons per second
- So, the algorithm takes about 499 seconds so more than 8 minutes!
- n $\log_2$ n for the n = 1,000,000 is about 20,000,000 which is about 0.02 seconds!

  (well there are some multiplicative constants that I ignore here but this is close)

HOW ABOUT THE "GENERAL" CASE

- Now the "cute" result: If we uniformly at random chose a pivot (each element that the same probability of being the pivot) then we can prove (see Section 6.10) that the

  expected run-time is proportional to n log n.

  How do we pick a pivot uniformly at random? (see class)

- So, with a good choice of pivot and a good implementation, QuickSort flies.

3. COUNTING

(We skip Chapter 2 for now)

Product Rule

This is a very important rule to know. Many useful applications.

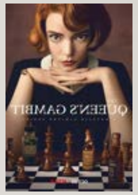
PRODUCT RULE

Let a procedure consists of performing a sequence of m tasks in order. Furthermore, assume that, for each i = 1, 2, …, m there are N_i ways to do the i-th task, regardless of how the first i-1 tasks were done. Then, there are

$N_1 * N_2 \ldots * N_m$ ways to do the entire procedure.

EXAMPLE

- you want to watch a movie on Netflix and order a meal.
- How may different ways (a movie, and a meal) are there for your evening?
- E.g., One way is to watch the Queen’s Gambit and order pizza. Another is to watch Extraction and order Chinese food.



- Let us say that you can watch one of 4,000 Netflix movies here in Canada and that you have a choice of 20 different take-out options.
- The theorem would tell us that you have 80,000 different options for your evening.

REMARK

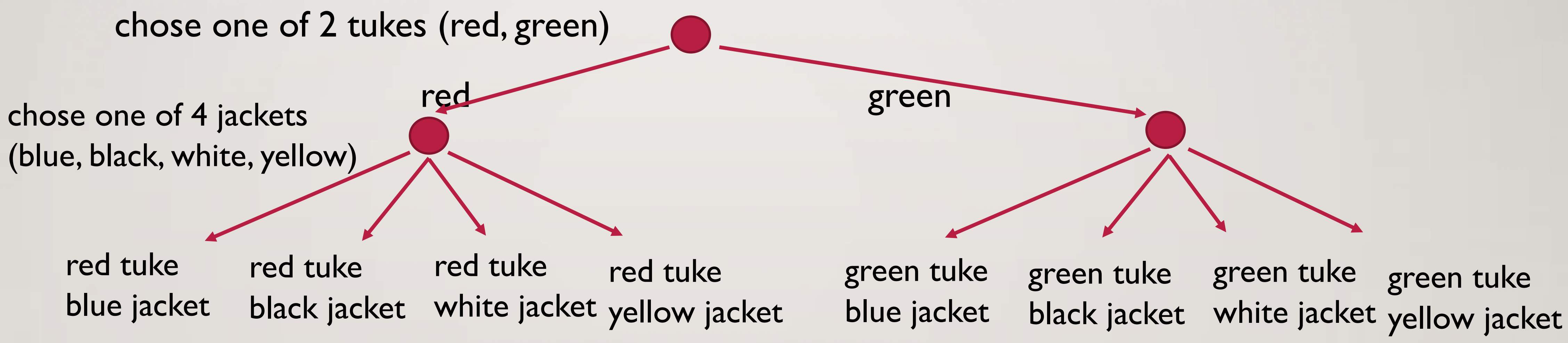
- Are the choices independent?   YES
- we can order a meal independently of which movie we have chosen
- So, N_1 is 4,000   and N_2 is 20
- The entire procedure consists of first choosing a movie then a meal.
- $N_1 * N_2 – 4{,}000*20=80{,}000$

ANOTHER EXAMPLE AND SOME EXPLANATION

- you have 2 tukes and 4 jackets. Let us say that you first chose a tuke and independently then a jacket. How may different outfits do you have consisting of one tuke and one jacket?

- The Product Rule says $N_1 = 2$, $N_2 = 4$, the two tasks are independent thus we have $2*4 = 8$ different outfits. (or ways to do the entire procedure of choosing a tuke and then a jacket)

ILLUSTRATION

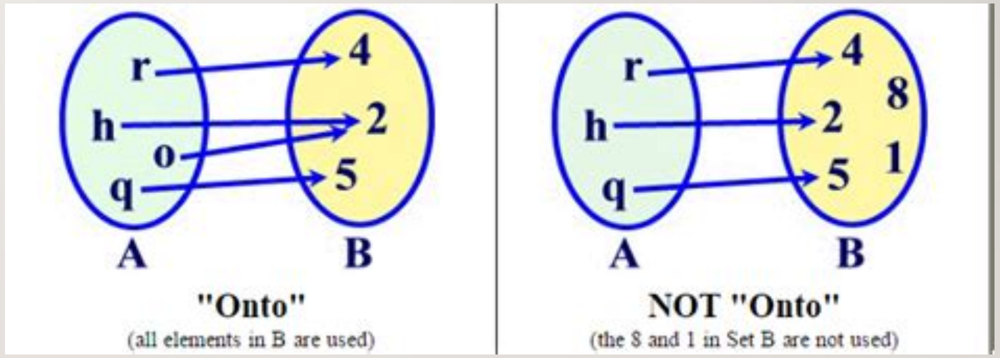


IMPORTANT NOTES

- Any outcome must be creatable in exactly one way.
  - “one-to-one”
  - In the tree representation this can be seen as each leaf can be reached by exactly one path
- All outcomes must be possible
  - “onto”
  - In the tree representation each of the outcomes is one of the leaves

RECALL FROM FIRST YEAR

- A function $f : A \rightarrow B$ is one-to-one if for any i and j with $i \neq j$ we have $f(a_i) \neq f(a_j)$.
- When is a function onto?


r
h
o
q
4
2
5
A
B
"Onto"
(all elements in B are used)
r
h
q
4
8
2
5
1
A
B
NOT "Onto"
(the 8 and 1 in Set B are not used)

F1:010100011
F2:011010100
F3:011010101
F4:110010100
F5:001100010

COUNTING BIT STRINGS

Let n≧1 be an integer. A bitstring of length n is a sequence of n 0's and 1's.

Example: n = 3   000, 0001, 010, 011, 100, 101, 110, 111

Question: How many bitstrings of length n are there?

We could guess likely the correct answer, but what is a proof for this?

The **procedure**: write the bitstring

for i = 1,2,…, n the i-th task is "write one bit".

The i^{th} task is independent on the previous tasks.

COUNTING BIT STRINGS CONT'D

- Two different executions of the procedure create two different bit strings.
- Each bitstring of length n can be created by this procedure.

- So, we can apply the Product Rule.
  - Each task i can be done in 2 ways. So, $N_i = 2$.
  - $N_1 * N_2 * \dots * N_n = 2^n$.

Result: There are 2^n bitstrings of length n.

COUNTING FUNCTIONS

- Let $m \geqq 1$ and $n \geqq 1$ be integers. Let A be a a set of size m and let B be a set of size n. How many functions $f : A \to B$ are there?
- Can we use the product rule?
- We should try to specify a procedure that generates all possible such functions.
- We thus need to specify the values of $f(a_1), f(a_2), \ldots, f(a_m)$

PROCEDURE

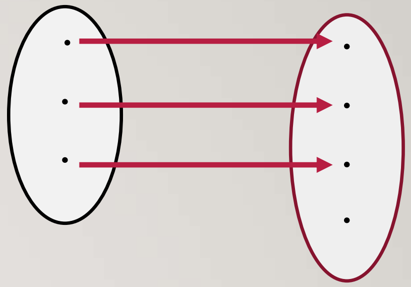
- For i = 1,2, …, m the **i-th task** is "specify the value of $f(a_i)$"
  - There are $N_i = n$ ways to execute the i-th task why?
  - The i-th task is independent on the i-1 earlier tasks.
  - Now apply Product Rule and get that there are $N_1 * N_2 \ldots * N_m = n*n*\ldots* = n^m$ ways to execute the entire procedure.

We therefore obtain the following theorem:

THEOREM – COUNTING THE NUMBER OF FUNCTIONS

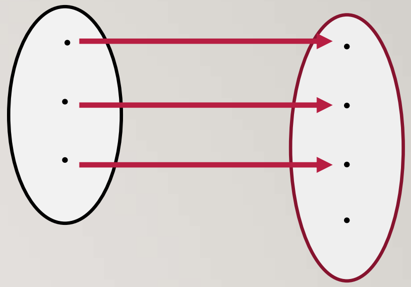
- **<u>Thereom:</u>** Let $m \geqq 1$ and $n \geqq 1$ be integers. Let A be a a set of size m and let B be a set of size n. The number of functions $f : A \rightarrow B$ is equal to $n^{m.}$

COUNTING ONE-TO-ONE FUNCTIONS



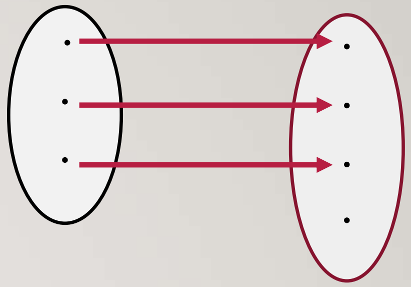
- How many one-to-one functions $f : A \rightarrow B$ are there?
- That depends on the relative sizes of A and B. Let again $|A| = m$ and $|B| = n$.
- What happens in case $m > n$? (see class)
- Assume now that $m \leqq n$.

COUNTING ONE-TO-ONE FUNCTIONS



- again
- For i = 1,2, …, m the **i-th task** is "specify the value of $f(a_i)$"
- however now there is a difference. Once we use in a step one of the elements of B we cannot reuse it anymore. Why?
- So, there are $N_1 = n$ ways to determine $f(a_1)$.
- For the second $f(a_2)$ we have one less element. ($B \setminus \{f(a_1)\}$) so n-1, thus $N_2 = n-1$.
- …

COUNTING ONE-TO-ONE FUNCTIONS



- For the i-th task: determine $f(a_i)$, we have the $B \setminus \{f(a_1), f(a_2), \ldots, f(a_{i-1})\}$ elements to chose from. Thus, there are $N_i = n-i+1$ to do the i-th task regardless of how we do the first i-1 tasks.
- By Product Rule, we get
  - There are $N_1 * N_2 \ldots * N_m = n(n-1)(n-2)\ldots(n-m+1)$ ways to do the entire procedure.
- Recall: $k! = \begin{cases} 1 & \text{for } k=0 \\ 1*2*\ldots*k & \text{if } k \geqq 1 \end{cases}$