Competitive Programming 234901

Introduction

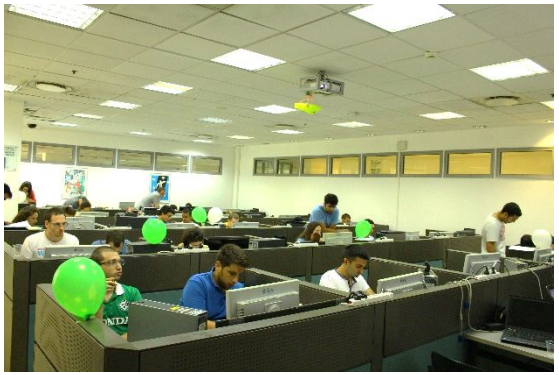
Teaching Language?

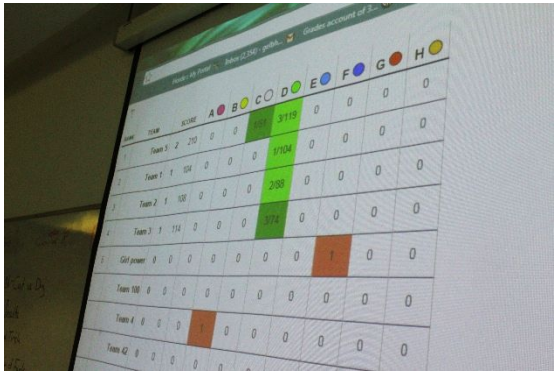
Agnda

- Workshop Goals
- Overview
  - Class Work
  - Homework
  - Final Competition
- Introduction to ACM-ICPC
- Warm-up problems

break

- Warm-up competition





Workshop Goals

Introduction to competitive programming

Assemble a team for the ACM-ICPC regionals

- Get hands-on experience
- Develop intuition
- Sharpen your programming and team-work skills
- Have fun!

Administration

- 3 Academic Points
- Prerequisites
  - Data Structures 1
  - Algorithms 1
  - (Or equivalent courses in other departments)
- ”Reshima Alef”
  - The course is not considered as a project or seminar
- Manual registration
- Tuesdays 16:30-19:30, Online

Course Staff

Prof. Gill Barequet
Lecturer in charge

Gil Ben-Shachar
TA in charge

Stav Perle
TA

Evgenii Zheltonozhskii
TA

Grading

- Final grade:
  - Class work
  - Homework (3 assignments)
  - Final Competition
- Attendance is mandatory
  - Missing one lesson will be excused
  - Additional absences (not including Miluim) will result in a penalty
- Full details: webcourse → information sheet

Class Work & Assigments

- Singles → Team up

Choose your teammates wisely!

- Grading: only number of solved problems
- Submission: vjudge + webcourse
- Bonus for late submission: ¼ problem in class

Estimated Schedule

HW0	Introduction
	Data Structures
HW1	Problem Solving Paradigms
	Pesach
	Math
	Graphs
	Geometry
	Work style
	Strings
HW2	More DP
	More Geometry
	Sponsors Event (?)
	More Graphs
	Mini competition
	Final competition

Using Zoom

- What if I have a question?
- During the lecture part:
  a. Raise your hand
  b. We will approach you
  c. Unmute yourself and ask it
- During the competition part:
  a. Raise your hand
  b. A TA will add you to a “breaking room”
  c. Ask the question in there

DO NOT CHEAT!

Final Competition

- Friday, 03/07/2020 (Depends on the covid situation)
- attendance mandatory!
- Format: similar to ACM ICPC.
- The best eligible team will be invited to represent the Technion at the ACM ICPC Southwestern Europe Regionals (SWERC)
  - SWERC 2020: Europe, November/December.
  - Travel expenses: covered by the computer science department and our sponsors.
  - All team members must be eligible.

ACM ICPC

- **ACM-ICPC** is an annual multi-tiered competitive competition among the universities of the world.

ACM ICPC

- The competition is running since 1977 and considered as the most prestigious in the world of competitive programming.

ACM ICPC

- The ACM-ICPC Competition has three tiers:

Local Qualification

- Each university selects teams for the regionals.
- Over 300,000 students throughout the world.

Regionals

- 30 different regions. The Technion competes in the "southwestern europe" region (SWERC).

World Finals

- Top 100 teams (2-4 from each region)

ACM ICPC Regionals

Previous Competitions

In 2020 the Technion is going to participate for the 15th time.

Previous results:

- 2006 Romania – Marcello Taub, Michael Gelfand, Kolman Vornovitsky – 30th
- 2007 Romania – Gill Cohen, Lior Biran, Yaron Yura – 36th place
- 2008 Romania – Shachar Papini, Yaniv Sabo, Karmi Grushko – 17th place
- 2009 Spain – Shachar Papini, Yaniv Sabo, Oshri Adler, Itai Levy – 7th place (Silver)
- 2010 Spain - Shachar Papini, Yaniv Sabo, Karmi Grushko – 2nd place (Gold)
- 2011 Spain – Omer Tabach, Tomer Fruman, Noa Korner – 16th place
- 2012 Spain – Idan Elad, Olivier Hoffman, Mike Harris – 12th place (Bronze)
- 2013 Spain – Alex Goltman, Oleg Zlotnik, Roei Gelbhart – 10th place (Bronze)
- 2014 Portugal – Eden Saig, Nofar Carmeli, Ori Brusilovsky – 15th place
- 2015 Portugal – Omer Daniel, Aviram Magen, Itay Zukier – 18th place
  Nitzan Tur, Ido Hakimi, Dima Kuznetsov – 38th place
- 2016 Portugal – Omer Daniel, Aviram Magen, Itay Zukier – 15th place
- 2017 Paris – Dean Leitersdorf , Vova Polosukhin, Artem Shtefan– 6th place (Silver)
- 2018 Paris – Dean Leitersdorf , Vova Polosukhin, Artem Shtefan– 24th place
  Ilan Ben Danan, Shira Weiss, Evgenii Zheltonozhskii- 39th place
- 2019 Paris – Evgenii Zheltonozhskii, Vova Polosukhin, Artem Shtefan– 11th place (Bronze)
  Ron Hirsch, Michael Zaloziecki, Rui Jiang - 47th place

SWERC 2019

https://swerc.eu

Competition format

- Teams of three
- One computer for each team
- 8-10 problems
- 5 hours
- Limited reference materials

- Automatic Judge

Competition Format (cont.)

- Programming languages: C, C++, Java
  - We will use C++ exclusively
- Code editor of your choice
  - Eclipse, Codeblocks, vim, emacs

- The focus is on **efficient**, **correct** solutions.

Judge System

- During the competition, solutions are submitted to the automatic judge system and checked against predefined test cases.
- Run time and memory use are usually very limited.
- After submitting the solution, the following answers are possible:
  - **ACCEPTED**
  - COMPILE-ERROR
  - TIME-LIMIT-EXCEEDED
  - RUNTIME-ERROR
  - WRONG-ANSWER
  - PRESENTATION-ERROR

ICPC Scoring and Ranking

- Teams are ranked by the **number of problems** solved.
- **Tie breaker**: Teams that solved the same amount of problems are ranked according to their *total time.*
  - Total time is the sum of problem acceptance times + 20 points of penalty for each unaccepted submission (only for problems that were eventually accepted)
- Example:
  - Total time=
    +40 (Acceptance time of A)
    +20 (Bad submission of A)
    +70 (Acceptance time of B)
    +20 (Bad submission of B)
    = 150

Time	Problem	Judge Answer
20min	A	No
40min	A	Yes
65min	B	No
70min	B	Yes
95min	C	No

ACM ICPC – Eligibility

Active student in the Technion next Winter semester?
(Ineligible if answer is no)

Willing and able to participate?
(Ineligible if answer is no)

first year of postsecondary studies >= 2016?
(Eligible if answer is yes)

Born in 1997 or later?
(Eligible if answer is yes)

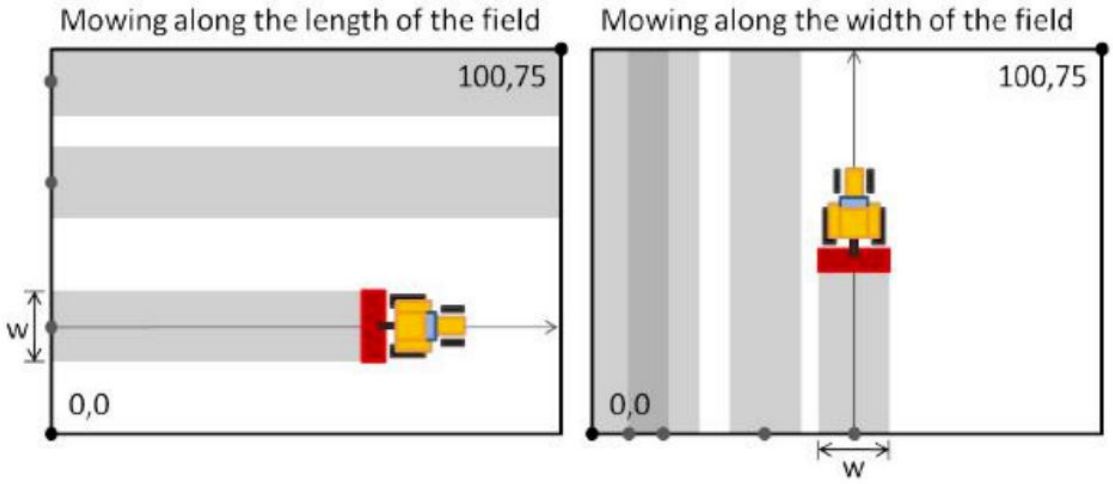
More than 8 semesters of full-time study?
(Ineligible if answer is yes)

https://icpc.baylor.edu/download/regionals/rules/EligibilityDecisionTree-2019.pdf

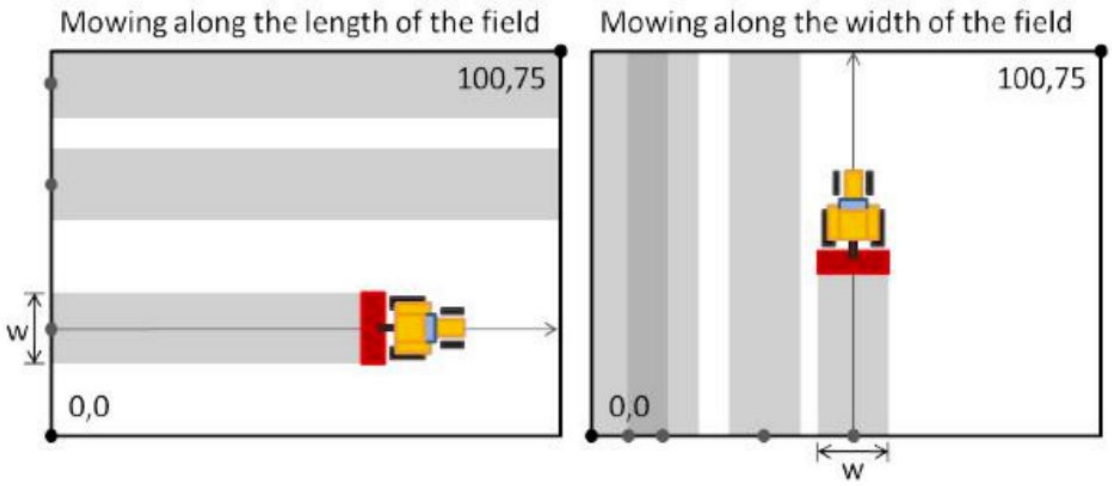
Example Problems

Example: Lawn Mower 4954

- The grass field in a the stadium is 100m long, 75m wide.
- The grass is mowed every week with special lawn mowers, always using the same strategy:
  - First, they make a series of passes along the length of the field.
  - Then they do the same along the width of the field.
  - All passes are straight lines, parallel to the sides of the field.



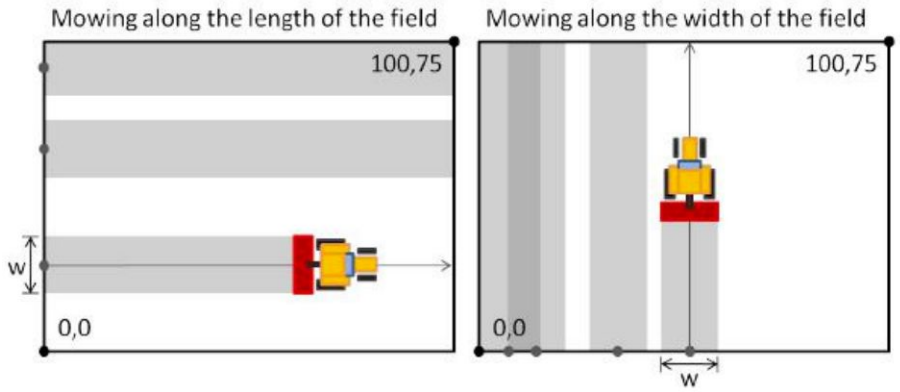
Lawn Mower 4954 (cont.)



- The new gardner likes to choose random starting positions for each of his passes.
- He is afraid of not doing a good job and being fired, so he has asked you to help him.

Lawn Mower 4954 (cont.)

- Write a program to make sure that the grass in the field is perfectly cut:
  - All parts of the field have to be mowed at least once when the mower goes from end to end, and again when the mower goes from side to side.
  - The grass field in a the stadium is 100m long, 75m wide.



Lawn Mower 4954 – Input

- Input:
  - Each test case contains 3 lines. The first line contains two integers: n_x $(0 < n_x < 1000)$ and n_y $(0 < n_y < 1000)$, and a real number w $(0 < w \leq 50)$, which represents the width of that particular lawn mower.
  - Next line contains n_x real numbers x_i $(0 \leq x_i \leq 75)$, describing the starting positions of the mower's center in the top-to-bottom passes.
  - Last line describes the side-to-side passes, with n_y real numbers y_i $(0 \leq y_i \leq 100)$.
  - The end of the test cases is signalled with a line that contains the numbers '0 0 0.0'.
- Output:
  - "YES" if all the grass was covered by the gardener (at least once in each direction) and "NO" otherwise.

Lawn Mower 4954 – Sample Input

- 8 11 10.0
  0.0 10.0 20.0 30.0 40.0 50.0 60.0 70.0
  0.0 10.0 20.0 30.0 40.0 50.0 60.0 70.0
  80.0 90.0 100.0
  (Output: YES)
- 4 5 20.0
  60.0 10.0 30.0 50.0
  30.0 10.0 90.0 50.0 70.0
  (Output: NO)

Lawn Mower 4954

- Any ideas?

Lawn Mower 4954 – Solution

- Solution
  - Check x and y axis separately.
  - Axis is covered iff the distance between all adjacent starting points is less than or equal to w.
  - The distance from the first/last point to the edge should be less or equal to $w/2$.
  - **Sort** the starting points, and go over them once.
- Time complexity: $O\left(n_x \log n_x + n_y \log n_y\right)$
  - $0 < n_x, n_y < 1000 \Rightarrow$ Running time is good enough! ☺

```
#include <iostream>
#include <algorithm>

#define MAX 1001

using namespace std;

int main()
{
  int nx, ny, i;
  double w;
  double x[MAX], y[MAX];
  while (cin >> nx)
  {
    cin >> ny >> w;
    if ((nx == 0)&&(ny == 0)
                  &&(w == 0))
      break;
    for (i = 0; i < nx; ++i)
      cin >> x[i];
    for (i = 0; i < ny; ++i)
      cin >> y[i];
    sort(x, x+nx);
    sort(y, y+ny);
    if ((x[0]>w/2)||(x[nx-1]<75-w/2)||
       (y[0]>w/2)||(y[ny-1]<100-w/2))
        {
            cout << "NO" << endl;
            continue;
        }
        for (i = 0; i < nx-1; ++i)
        {
            if (x[i+1]-x[i] > w)
            {
                cout << "NO" << endl;
                break;
            }
        }
        if (i != nx-1)
            continue;
        for (i = 0; i < ny-1; ++i)
        {
            if (y[i+1]-y[i] > w)
            {
                cout << "NO" << endl;
                break;
            }
        }
        if (i != ny-1)
            continue;
        cout << "YES" << endl;
    }
    return 0;
}
```

Conclusions from the Code

- Input/Output should be as simple as possible
- No need to check for input errors
- Pay attention to edge cases!
- Feel free to copy and paste
- Simple variable names

- Sometimes you do have to refactor your code and solve bugs. Look for the sweet spot between quick writing and code clarity.

Complexity in the Competition

- In the problem statements we are given input bounds, and from them we can deduce if our expected running time is good enough.
- Usually the code has only 2-3 seconds to run in each submission.
- **Rule of thumb**: ~10M operations is the bound above which a solution is not feasible.
  - For example, time complexity of $O(N^2)$ is feasible for $N = 3000$, but not for $N = 10000$.
  - This rule is good in most cases, but not in all of them. Sometimes N^2 gets accepted and $10N^2$ doesn't.

Additional Materials

- In ACM-ICPC, contestants are only allowed to use C++ and the STL data structures (or Java and the standard library)
- The official documentation of C++ and STL is given in the competition.
- In addition, you are allowed to bring up to 25 pages of printed materials.
  - A good reference sheet is very useful in the competition!
  - We recommend starting to gather materials now.

6823 - Counting Substhreengs

- Substrings are strings formed by choosing a subset of contiguous characters from a string (This is well known)
- A *Substhreeng* is a substring which complies to the following additional requirements:
  - It is non-empty, and composed entirely of base 10 digits.
  - Interpreted in base 10 (allowing extra leading zeros), the resulting integer is a multiple of 3.

Example

- The string ‘130a303’ contains 9 substhreengs:
  - 1**3**0a303
  - 1**30**a303
  - 13**0**a303
  - 130a**3**03
  - 130a**30**3
  - 130a**303**
  - 130a3**0**3
  - 130a30**3**
  - 130a3**03**

Input/Output

- The input contains several test cases.
- A test case consists of a single line that contains a non-empty string S of at most 10^6 characters.
- Each character of S is either a digit or a lowercase letter.

- For each test case output a line with an integer representing the number of substhreengs contained in S.

Input/Output

130a303	9
0000000000	55
icpc2014regional	2

6823 - Counting Substhreengs

Any Ideas?

How Can We Solve It?

- We define: $n = 10^n d_n + 10^{n-1} d_{n-1} + \cdots + d_0$
- For numbers divisible by three:

$$(n \; mod \; 3) = 0 \Leftrightarrow \left(\sum_{i=0}^{n} d_i \; mod \; 3 \right) = 0$$

20144

Original Number 2 0 1 4 4

20144

Original Number		2	0	1	4	4
Prefix Sum	0	2	2	3	7	11
Prefix Sum mod 3	0	2	2	0	1	2

$$\underbrace{\binom{3}{2} = 3}_{sum=2} \qquad \underbrace{\binom{2}{2} = 1}_{sum=0} \qquad \underbrace{\binom{1}{2} = 0}_{sum=1}$$

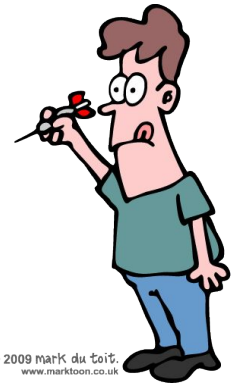
$$3 + 1 + 0 = 4$$

```
int main() {
     string S;
     while(cin>>S) {
          S += "$";
          long long total_substhreengs=0;
          vector<long long> hist(3,0);
          hist[0]=1;
          int digit_sum=0;

          for(unsigned int i=0; i<S.size(); ++i) {
               int num = char(S[i])-'0';
               if ((num>=0) && (num<=9)) {
                    digit_sum += num;
                    digit_sum %= 3;
                    hist[digit_sum]++;
               } else {
                    for(int k=0; k<3; k++) {
                         total_substhreengs += hist[k]*(hist[k]-1)/2;
                    }
                    hist=vector<long long>(3,0);
                    hist[0]=1;
                    digit_sum=0;
               }
          }

          cout << total_substhreengs << endl;
     }
     return 0;
}
```

4507 – Darts

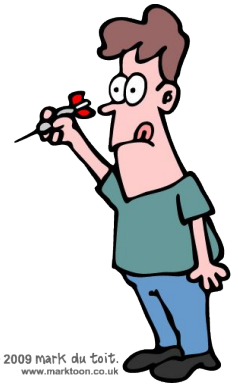
- Players start with a score of N points (typically, N = 501) and take turns throwing darts.
- The score of each player decreases by the value of the section hit by the dart, unless the score becomes negative, in which case it remains unchanged.
- The first player to reach a score of 0 wins.

4507 – Darts



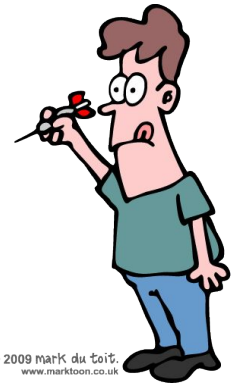
- Player A throws the darts at random, and consequently they land with equal probability in each of the sections of the dartboard.
- If Player B aims at a certain section, the dart has the same probability of landing in the correct one as in each of the two adjacent ones (the neighboring regions to the left and right).
  - Moreover, he is completely aware of his ability and sober enough to aim at the section that maximizes his probability of winning

4507 – Darts



- Given the initial score of both players, can you determine the probability that the first player wins? Of course, being the first to throw a dart might be advantageous, so the answer depends on who plays first.

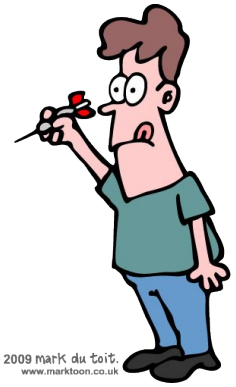
4507 – Darts



- $d_i = 20,1,18,4, \ldots$ Score of each sector
- $p_A(n,m) =$ Probability of A win if it's A's turn.
  - $n =$ Score of A
  - $m =$ Score of B
- $p_B(n,m) =$ Probability of B win if it's B's turn.

- **Problem**: compute $p_A\ (N,N)$ and $p_B(N,N)$.
- **Solution**: Dynamic Programming

Recurrence

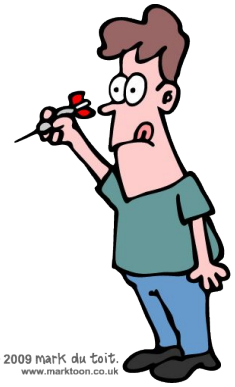


- $p_A(0,m) = 1$
- $p_B(n,0) = 1$

- $p_A(n,m) = \frac{1}{20}\sum_{i=1}^{20}\left(1 - p_B(n-d_i,m)\right)$
- $p_B(n,m) = \max_{1\le i\le 20} \frac{1}{3}\sum_{j=-1}^{1}\left(1 - p_A\left(n,m-d_{i+j}\right)\right)$

- Only valid if $n \geq 20$ and $m \geq 20$!

Solving for $n, m \leq 20$

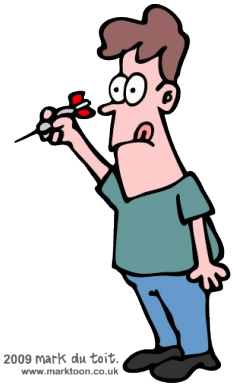


- When $n, m < d_i$ we get a cyclic dependency:

$$p_A(n,m) = \frac{1}{20}\sum_{i=1}^{20}\big(1 - p_B(n - d_i, m)\big) = \cdots = A + B \cdot p_B(n,m)$$

$$p_B(n,m) = \cdots = C + D \cdot p_A(n,m)$$

Solving for $n, m \leq 20$



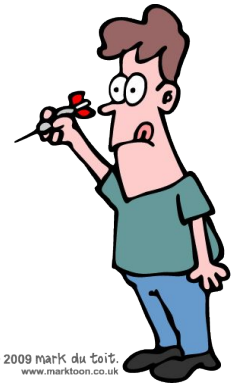
- 2 equations in 2 variables:

$$p_A(n,m) - B \cdot p_B(n,m) = A$$
$$-D \cdot p_A(n,m) + p_B(n,m) = C$$

$$\begin{pmatrix} p_A^{nm} \\ p_B^{nm} \end{pmatrix} = \begin{pmatrix} 1 & -B \\ -D & 1 \end{pmatrix}^{-1} \begin{pmatrix} A \\ C \end{pmatrix}$$

- B maximizes winning probability:
  - Check different numbers and solve.
  - Take best solution.

Notes



- **Time Complexity**: $O(N^2) \underset{N \leq 501}{\approx} 10^6$

- We can prepare one 501×501 table and use it for all test cases.

- Don’t forget std::setprecision