CS-E4590
Competitive Programming

Autumn 2017

Jukka Suomela
Aalto University

Goals

- Learning to ***program well in practice***
- Small, self-contained complete programs:
  - read input
  - solve a ***computational problem***
  - print output

Goals

- This course:
  - designing ***efficient algorithms*** that you can ***implement*** correctly & quickly
- Other courses:
  - best practices in large-scale software development
  - algorithms that are good in theory but not in practice
  - parallelism, large-scale high-performance computing

Goals

- Challenging but fun
- Good practice for programming contests
  - **NCPC: 7 October 2017** (ncpc.aalto.fi)
- Learning about the “big picture”
  - new connections between familiar concepts

Level

- Even if you are a 1st year student, you will be able to ***solve some problems***
- Even if you have a PhD in CS, you will learn ***something new about algorithms***

What is needed?

- ***Lots of practice!***
  - practicing ***programming***: you will get basic things right automatically, without much thinking
  - practicing ***problem solving***: you will gain intuition on what kind of algorithm design techniques typically work in different problems
- Some theoretical knowledge on algorithms

Practicalities

- Read ***MyCourses***
- Join Slack: **aaltocontests.slack.com**
- Register in CSES: **cses.fi**
- Configure ***full real name + email address***

Practicalities

- If interested in credits:
  read information in MyCourses!
  - **2 credits**: active participation in at least 5 meetings, at least 1 problem solved correctly each time
  - **more credits**: ask me!
- If interested in learning:
  lots of self-study expected!

Practicalities

- Typical meetings:
  - **1h** lecture
  - **3h** practice contest
  - **1h** discussion
- Homework

This week

- Graphs
  - concepts, representations
- Flows and cuts in graphs
  - ***max flow = min cut***
- Applications
  - using max-flow algorithms as subroutines

Key concepts

- Graph, node, edge
  - weighted graph, directed graph
- Path
  - length of path, shortest path, distance, connectivity
- Cycle
  - length of cycle, acyclic graph, tree

Key concepts

- Depth-first search (DFS)
- Breadth-first search (BFS)

Key concepts

- Flow
  - maximum flow
- Cut
  - minimum cut

Key concepts

- Bipartite graph
- Matching
  - maximum matching, maximum-weight matching
  - perfect matching

Algorithmic ideas

- Find augmenting paths until you get stuck
- Variants:
  - find ***any*** path (DFS), “**Ford–Fulkerson**”
  - find ***heavy*** path (DFS), “**scaling**”
  - find ***short*** path (BFS), “**Edmonds–Karp**”

Standard notation

- Graph $G = (V, E)$, $n = |V|$, $m = |E|$
  - node $v \in V$
  - edge $e \in E$
  - undirected edge $\{u, v\} \in E$
  - directed edge $(u, v) \in E$
  - source $s \in V$, sink $t \in V$

Representations: undirected

- `x[i]` = list of neighbours of node *i*
  - *O*(*m*) space, e.g. vector of vectors
- `x[i][j]` = does there exist edge {*i*, *j*} ?
  - *O*(n^2) space, symmetric square matrix
- list of edges, set of edges …

Representations: directed

- `x[i]` = list of successors of node *i*,
  `y[i]` = list of predecessors of node *i*
  - *O*(*m*) space, e.g. vector of vectors
- `x[i][j]` = does there exist edge (*i*, *j*) ?
  - $O(n^2)$ space, square matrix
- list of edges, set of edges …

Representations: weighted

- `x[i]` = list of edges incident to node *i*
  - e.g. list of (*j*, *w*) pairs
- `x[i][j]` = weight of the edge {*i*, *j*}
  - often “edge of weight 0” = “no edge”
- list of (*i*, *j*, *w*) triples, map (*i*, *j*) → *w* …

Representations: flow

- “Capacity” and “flow” can be represented just like any other weight
- Convenient: representations where easy to navigate between edges (i, j) and (j, i)
  - maintain “opposite edge” pointers?
  - store both of them together?

Representations

- One size does not fit all
- Often worth thinking a bit before choosing a representation
- Perfectly fine to convert between different representations

Next steps

- Help with homework problems
- Practice contest