Stable Matching

Data Structures and Algorithms
Andrei Bulatov

Stable Matching Problem: Formalism

A pair (m,w') is an instability with respect to a matching S if (m,w') does not belong to S, but both m and w' prefers each other to their current matches.

A matching is stable if it is (i) perfect, and (ii) has no instabilities

Problem

There are n men and n women with their preference lists.

(a) Does there exist a stable matching?

(b) If a stable matching exists, how can we find it?

We have Gale-Shapley algorithm to solve it

Stable Matching Problem: Analysis

Theorem

G.-S. algorithm returns a stable matching

Theorem

G.-S. algorithm terminates after at most n^2 iterations of the while loop

Brute Force

For many problems there is a very simple algorithm

For example, for the Stable Matching Problem we could try all perfect matchings

Difficulty: there are too many of them, n!

Such an algorithm is called a brute force algorithm:

enumerate all possible configurations and choose the right one

An efficient algorithm should outperform the brute force algorithm

- substantially
- provably / analytically

Polynomial Time

Good criterion of scaling: If the instance size is doubled the running time increases by a constant factor

Natural example: polynomials

Let the running time is $f(n) = 3n^d + 2n^2 + n$

When doubling the instance size

$$f(2n) = 3(2n)^d + 2(2n)^2 + 2n \le 2^d(3n^d + 2n^2 + n)$$

An algorithm has polynomial running time, or is a polynomial time algorithm if its running time is bounded from above by a polynomial

Is it good?

Polynomial Time (cntd)

Contras:

- there are bad polynomial time algorithms
- there are good non-poly time algorithms
- it does not capture the `practical' complexity of algorithms

Pros:

- usually if there is a poly time algorithm, there is a good one
- it captures something

Running time	Size n					
	10	20	30	40	50	60
n	.00001 seconds	.00002 seconds	.00003 seconds	.00004 seconds	.00005 seconds	.00006 seconds
n^2	.0001 seconds	.0004 seconds	.0009 seconds	.0016 seconds	.0025 seconds	.0036 seconds
n^3	.001 seconds	.008 seconds	.027 seconds	.064 seconds	.125 seconds	.216 seconds
n^5	.1 seconds	3.2 seconds	24.3 seconds	1.7 minutes	5.2 minutes	13.0 minutes
2^n	.001 seconds	1.0 seconds	17.9 minutes	12.7 days	35.7 years	366 centuries
3^n	.059 seconds	58 minutes	6.5 years	3855 centuries	2×10^8 centuries	1.3×10^{13} centuries

Asymptotics

We don't want to compute the exact running time

Why?

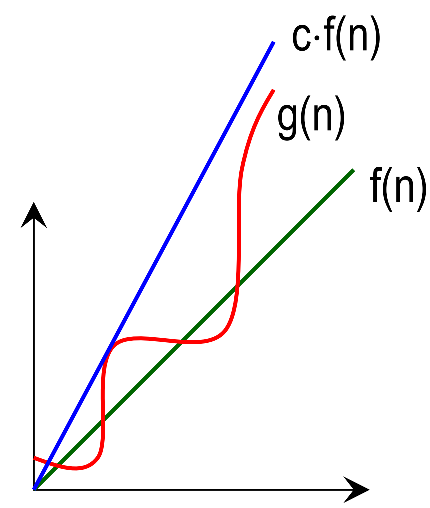
- Do we care if the running time is $2.53n^2 + 3.42n$ or $2.55n^2 + 3.39n$?
- We will mostly represent algorithms by pseudocode.
  Implementation details can change running time by some constant
- We are interested in more conceptual differences between algorithms, and will be happy with a rough classification

Then $2.53n^2 + 3.42n$ is more or less similar to n^2

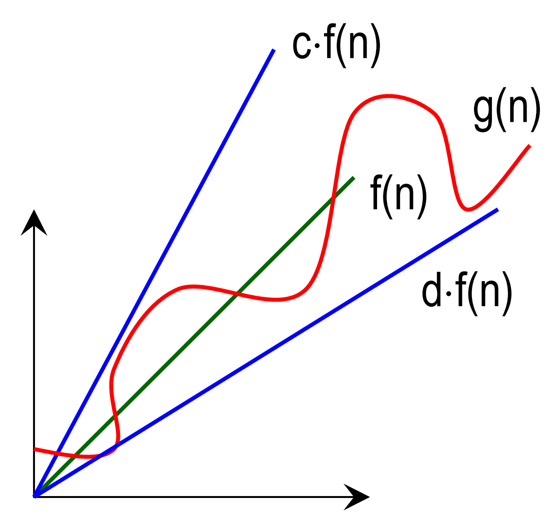
Asymptotic Notation

- For two functions f,g: $\mathbb{N} \to \mathbb{R}$

- g is in O(f) if there is c such that starting from some k: $g(n) \le c \cdot f(n)$

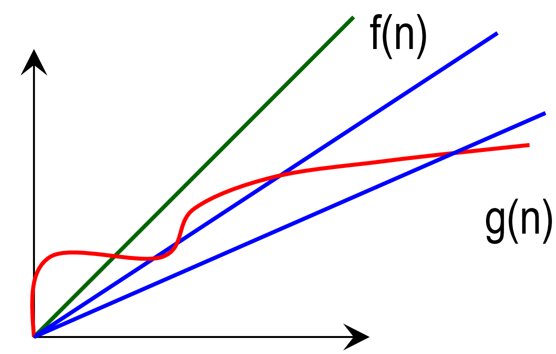


- g is in Θ(f) if there are c,d > 0 such that starting from some k: $d \cdot f(n) \le g(n) \le c \cdot f(n)$



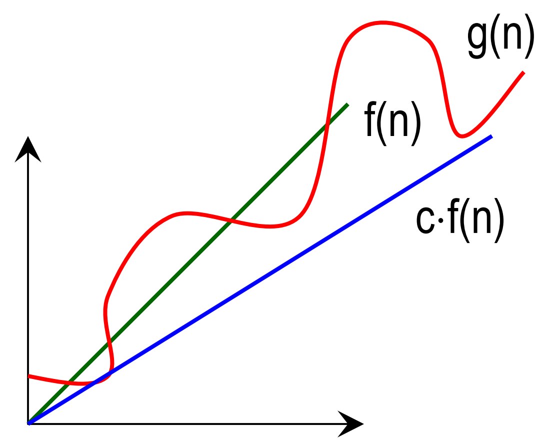
Asymptotic Notation

- g is in $o(f)$ if for any c starting from some $k(c)$: $g(n) < c \cdot f(n)$



- g is in $\Omega(f)$ if there is c such that starting from some k:

  $g(n) \geq c \cdot f(n)$



Read about asymptotic notation

Stable Matching Problem: Gale-Shapley Algorithm

Input: sets M and W of men and women with preference lists

Output: a stable matching

```
initially all m∈M and w∈W are free
while there is a free man do
  let w be the highest ranked woman for m to whom he
   hasn't yet proposed
  if w is free then (m,w) become engaged
  else if w is currently engaged to m'
     if w prefers m' to m then m remains free
     else w prefers m to m'
         (m,w) become engaged
         m' becomes free
     endif
  endif
endwhile
Return the set S of engaged pairs
```

Implementation: Choosing Data Structures

The choice of data structures is determined by what we have to do with data

Ideally, every operation with data should take constant time

For Stable matching we need:

- to identify a free man
- for a man m to indentify his highest ranked woman he hasn’t proposed
- for a woman w to decide if w is engaged, and if yes who is her current partner
- for a woman w and two men m and m' to decide, whom w prefers

Arrays

A list of elements:

- fixed length
- direct (constant time) access to element # i

Array operations:

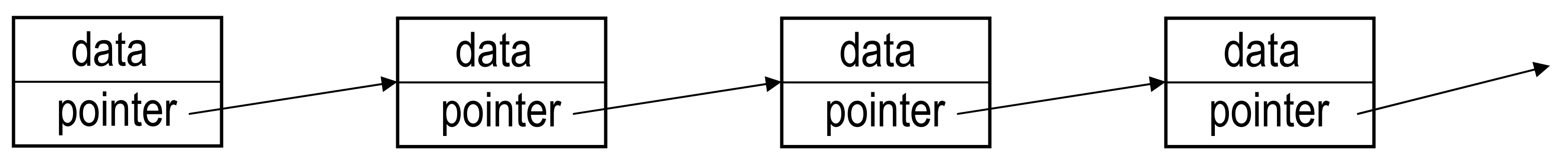
- read element # i: $O(1)$
- find a required element: $O(n)$ if unordered
  $O(\log n)$ if ordered
- insert an element: $O(n)$

Lists

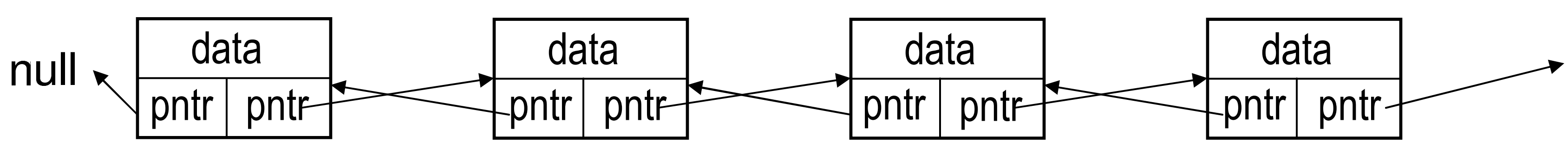
A list of elements:

- variable length
- no direct access to element # i

Linked list

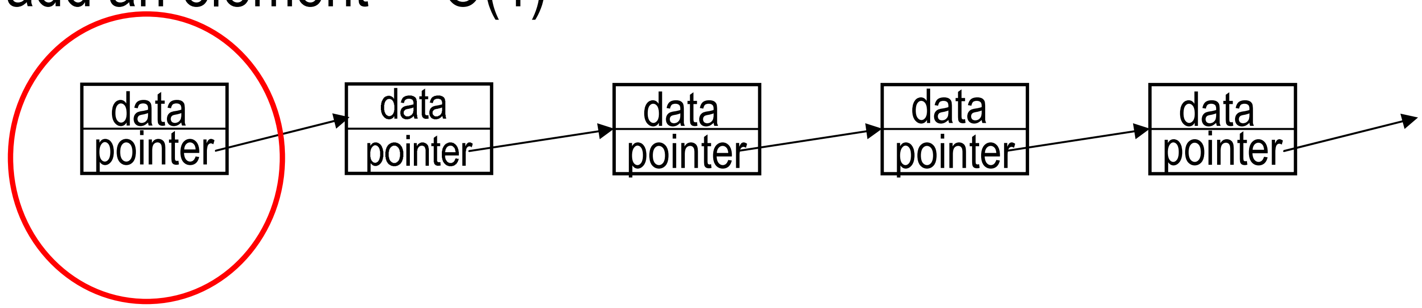


Double linked list

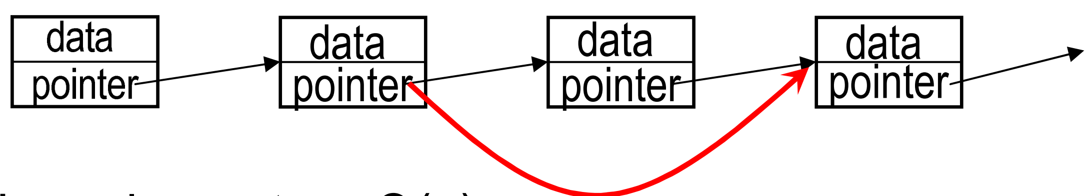


List Operations

- add an element O(1)



- delete element O(1)



- find an element O(n)

Data Structures for Stable Matching

Let M and W contain n elements each

- preference lists: arrays
  ManPref[m,i], WomanPref[w,i]
- next woman to propose: array
  Next[m]
- current partner of a woman: array
  Current[w]
- set of free men: linked list
  FreeMan
- ranking of men: array
  Ranking[w,m]

adding/finding/removing a free man O(1)

finding next woman to propose
ManPref[m,Next[m]] O(1)

deciding on woman's current partner O(1)

deciding whom a woman prefers
Ranking[w,m] < Ranking[w,m']
O(1)

Homework

Write pseudocode for G.-S. algorithm with data structures