Complexity & Analysis of Data Structures & Algorithms

Piyush Kumar

(Lecture 1: Introduction)

Welcome to COP4531

Based on slides from
J. Edmonds, S. Rudich, S. H. Teng,
K. Wayne and my old slides.

Today

- My Info : Timings for the class
- References
- Pre-Requisites Survey
- How you will be graded
- Syllabus
- About Advanced Algorithms
  - and its applications
- Our First Problem
  - Stable Matching

Instructor

Piyush Kumar

161 Love Building

Ph: 850-645-2355

Web page: http://piyush.compgeom.com

Office Hours: On course info sheet

Email:

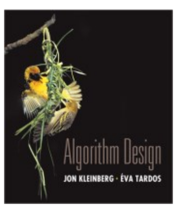
piyush at acm dot org

Class/Exam Timings

- Timings
  - See course handout/webpage
- Midterm:
  - See course handout/webpage
- Final Exam
  - See course handout/webpage

Other Details

- Course web site:
  - http://piyush.compgeom.com/teach/4531
- Textbook.



References

- Klienberg / Tardos
  - Algorithm Design
- Other References
  - [CLRS] T. Cormen, C. Leiserson, R. Rivest, and C. Stein. Introduction to Algorithms (2nd edition).
  - My slides and notes

PreReq

- Data Structures
- Introduction to Probability (STA 4442/STA 3032)
  - Any other substitute classes?
- C++ / [Python]
- Discrete Mathematics II (MAD 3105) or
  - Mathematics in Computing (MAD 3107)
- Basic Math skills
- Lots of Time…
- ToDo List:
  - Get a LinProg Account
  - Get a copy of the text book.

PreReq

- COP 4530 or higher

  (What this class does not cover)
  - Linked Lists, Stacks.
  - Binary Trees, Heaps.
  - STL, containers/iterators.
  - Mathematical Induction / Contradiction
  - Basic Probability/Expectations.

What can you expect?

- After the course expect to
  - Know more about algorithms (of course)
  - Think algorithmically
  - Know how to solve real world algorithmic problems
    - Both in theory (algorithm) and practice (code)
  - Be better at applications that require algorithms:
    - and apply algorithms to places you never imagined…

Grading*

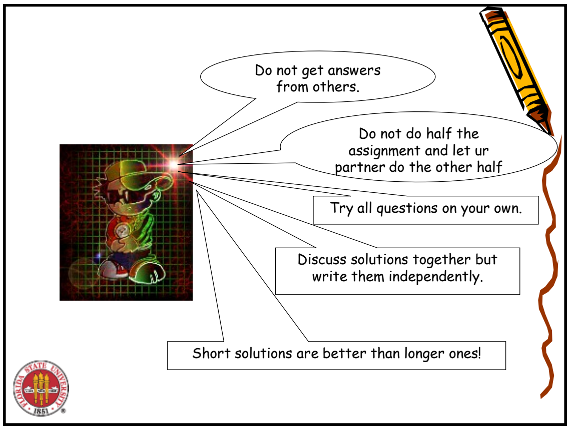
- Homework : 10%
- Programming Project: 15 %
- Class Participation : 5%
- Surprise Quizzes: 20%
- Midterm : 20%
- Final Exam : 30%

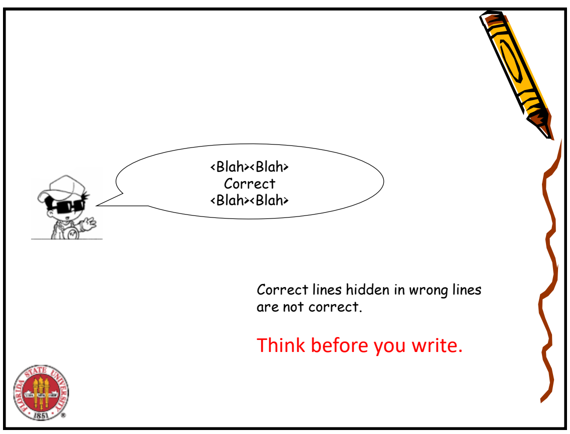
Designing Algorithms

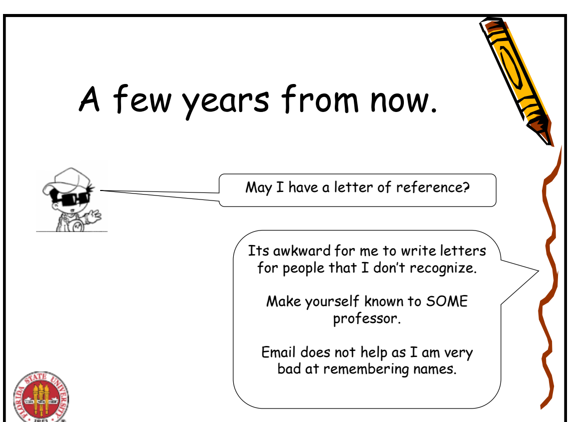
How to do well in this class?

Doing Well in this class.

- Study in Groups
- Assignments are done in pairs
- Also Learn from one another.


Do not get answers from others.
Do not do half the assignment and let ur partner do the other half
Try all questions on your own.
Discuss solutions together but write them independently.
Short solutions are better than longer ones!


<Blah><Blah>
Correct
<Blah><Blah>
Correct lines hidden in wrong lines are not correct.
Think before you write.


A few years from now.
May I have a letter of reference?
Its awkward for me to write letters for people that I don't recognize.
Make yourself known to SOME professor.
Email does not help as I am very bad at remembering names.

Expectations

- Work hard and learn/understand the material well.
- Feel free to ask questions.
- Take help from me and the TA.
- Course Load: 4-6 hours a week (assumes you are getting help).

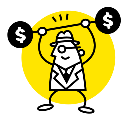
Thinking about Algorithms

Be Creative

- Ask questions
- Why is this done this way and not that way?
- Guess potential methods to solve the problem
- Look for **counterexamples**.
- **Start Day dreaming**: Allow the essence of the material to seep into your subconscious.

Boss assigns task.

- Given today's prices of pork, grain, sawdust, …
- Given constraints on what constitutes a hotdog.
- Make the cheapest hotdog



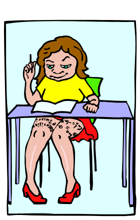
Everyday industry asks these questions.

Your Answer:

- Tell me what to code.

With more suffocated software engineering systems, the demand for mundane programmers will diminish.

Your Answer:



- I learnt this great algorithm that will work.

Soon all known algorithms will be available in libraries

Your answer:

- I can develop a new algorithm for you.

Great thinkers will always be needed.

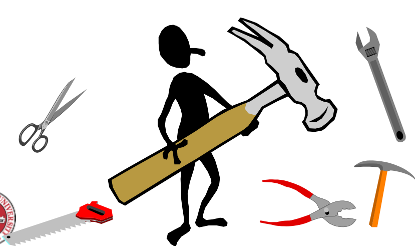
Course Content

- A list of algorithms.
  - Learn their code.
  - Trace them until you are convinced that they work.
  - Implement them.
  - Worry about details.

```
class InsertionSortAlgorithm : public SortAlgorithm
{
 void sort(int a[]) {
     for (int i = 1; i < a.length; i++) {
       int j = i;
       int B = a[i];
       while ((j > 0) && (a[j-1] > B)) {
         a[j] = a[j-1];
         j--; }
       a[j] = B;
     }}
```

Course Content

- A survey of algorithmic design techniques.
- Abstract thinking.
- How to develop new algorithms for any problem that may arise.



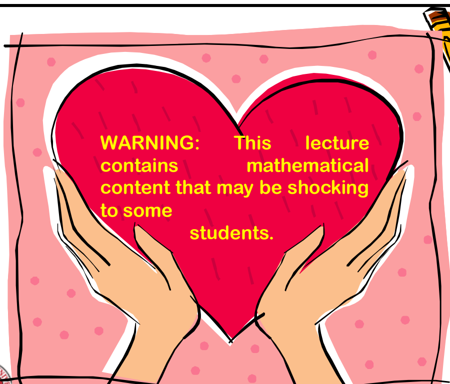
Syllabus*

- Asymptotic Analysis and Recursions
- Graph Algorithms
- Greedy Algorithms
- Divide and Conquer
- Dynamic Programming
- Network Flows
- Complexity Classes and Approximation Algorithms
- Computational Geometry
- Parallel Algorithms

* Tentative

Stable Marriage

Our first problem



The problem

- There are n men and n women
- Each man has a preference list, so does the woman.
- These lists have no ties.

- Devise a system by which each of the n men and n women can end up getting married.

Other Similar problems

- Given a set of colleges and students pair them. (Internship – Company assignments)
- Given airlines and pilots, pair them.
- Given two images, pair the points belonging to the same point in 3D to extract depth from the two images.
- Dorm room assignments.
- Hospital residency assignments**.

Stereo Matching

Fact: If one knows the distance between the cameras And the matching, its almost trivial to recover depth..

A Good matching/pairing

- Maximize the number of people who get their first match?
- Maximize the av?
- Maximize the minimum satisfaction?
- Can anything go wrong?

Example Preference Lists

Man	1st	2nd	3rd
X	A	B	C
Y	B	A	C
Z	A	B	C

Woman	1st	2nd	3rd
A	Y	X	Z
B	X	Y	Z
C	X	Y	Z

What goes wrong?

Unstable pairs: (X,C) and (B,Y)
They prefer each other to current pairs.

Stable Matching

Man	1st	2nd	3rd
X	A	B	C
Y	B	A	C
Z	A	B	C

Woman	1st	2nd	3rd
A	Y	X	Z
B	X	Y	Z
C	X	Y	Z

No Pairs creating ***instability***.

Another Stable Matching

Man	1st	2nd	3rd
X	A	B	C
Y	B	A	C
Z	A	B	C

Woman	1st	2nd	3rd
A	Y	X	Z
B	X	Y	Z
C	X	Y	Z

Stability is Primary.

- Any reasonable list of criteria must contain the stability criterion.
- A pairing is doomed if it contains a shaky couple.

Main Idea

Idea: Allow the pairs to keep breaking up and reforming until they become stable

Can you argue that the couples will not continue breaking up and reforming forever?

Men Propose (Women dispose)

```
Initialize each person to be free.
while (some man m is free and hasn't proposed to every woman)
    w = first woman on m's list to whom m has not yet proposed
    if  (w is free)
        assign m and w to be engaged
    else if  (w prefers m to her fiancé m')
        assign m and w to be engaged, and m' to be free
    else
        w rejects m
```

Gale-Shapley Algorithm (men propose)

Analysis

- Does the algorithm terminate?
- Running time?
- Space requirement?

Improvement Lemma

- Improvement Lemma: If a woman has a committed suitor, then she will always have someone at least as good, from that point in time onwards (and on the termination of the algorithm).

Corollary : Improvement Lemma

- Each woman will marry her absolute favorite of the men who proposed to her.

Demotion Lemma

- The sequence of women to whom m proposes gets worse and worse
  (in terms of his preference list)

Lemma 1

- No Man can be rejected by all the Women.
- **Proof: ??**

Contradiction

Suppose Bob is rejected by all the women.
At that point:
Each women must have a suitor other than Bob
(By Improvement Lemma, once a woman has a suitor she will always have at least one)
The n women have n suitors, Bob not among them.
Thus, there must be at least n+1 men !

Corollary: Lemma 1

- If m is free at some point in the execution of the algorithm, then there is a woman to whom he has not yet proposed.

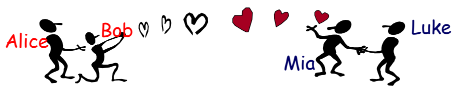
Corollary: Lemma 1

- The algorithm returns a matching. (Since no man is free?)
- The algorithm returns a perfect matching. (Since there is no free man?)

Lemma 2

- Consider the execution of the G-S algorithm that returns a set of pairs S. The set S is a stable matching.
- Proof?

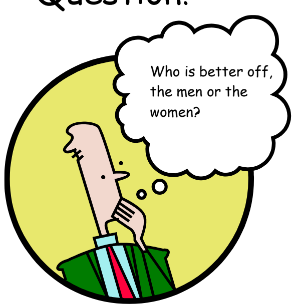
Lemma 2
Proof by contradiction



Unstable pair : Bob and Mia

- This means Bob likes Mia more than his partner, Alice.
- Thus, Bob proposed to Mia before he proposed to Alice.
- Mia must have rejected Bob for someone she preferred.
- By the Improvement lemma, she must like her parnter Luke more than Bob.

Question!



Best (Valid?) Parter for Bob?

- Best woman for “Bob”?
- The woman at the top of Bob’s list?

A woman w is a ***valid partner*** of a man m if there is a Stable matching that contains (m,w).
A man’s optimal match or ***best valid partner*** is the highest ranked woman for whom there is <u>some</u> stable pairing in which they are matched

She is the best woman he can conceivably be matched in a stable world. Presumably, she might be better than the woman he gets matched to in the stable pairing output by GS.

Example

- M { w, w' }
- M' { w' , w }
- W { m' , m }
- W' { m , m' }

Two stable matchings:
(m,w) (m',w')
Or (m',w) (m,w')

Worst Valid Partner Match.

- A Man's *worst valid partner* is the lowest ranked woman in his preference list that is a valid partner.

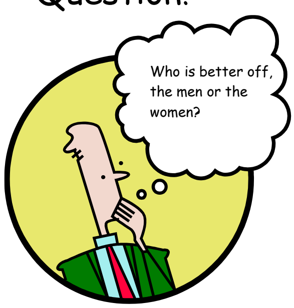
Dating Dilemma

- A pairing is man-optimal if every man gets his best valid partner. This is the best of all possible stable worlds for every man simultaneously.
- A pairing is man-pessimal if every man gets his worst valid partner. This is the worst of all possible stable worlds for every man simultaneously.

Dating Dilemmas

- A pairing is woman-optimal if every woman gets her best valid partner. This is the best of all possible stable worlds for every woman simultaneously.
- A pairing is woman-pessimal if every woman gets her worst valid partner. This is the worst of all possible stable worlds for every woman simultaneously.

Question!



Mathematical FACT.

The traditional marriage algorithm (a.k.a. G-S alg.) always produces a **man-optimal** and **woman-pessimal** pairing.

Theorem 1: GS Produces man-optimal pairing.

Theorem 2: GS produced pairing is woman-pessimal.

Theorem 1
Proof by contradiction

- Suppose not: That some man gets rejected by his best valid partner during the execution of GS.

(w.l.o.g. Let Bob be the first such man)

- Bob gets rejected by his optimal match Mia who says "maybe" to Luke (whom she prefers)
- Since Bob was the only man to be rejected by his optimal match so far, Luke must like Mia at least as much as his optimal match.

We are assuming that Mia is Bob's optimal match, Mia likes Luke more than Bob. Luke likes Mia at least as much as his optimal match.

- We now show that any pairing S in which Bob marries Mia cannot be stable (for a contradiction).
- Suppose S is stable:
  - Luke likes Mia more than his partner in S
    - Luke likes Mia at least as much as his best match, but he is not matched to Mia in S
  - Mia likes Luke more than her partner Bob in S

We are assuming that Mia is Bob's optimal match, Mia likes Luke more than Bob. Luke likes Mia at least as much as his optimal match.

- We've shown that any pairing in which Bob marries Mia cannot be stable.
  - Thus, Mia cannot be Bob's optimal match (since he can never marry her in a stable world).
  - So Bob never gets rejected by his optimal match in GS, and thus GS is man-optimal.

GS is woman-pessimal

- We know it is man-optimal. Suppose there is a GS stable pairing S* with (Luke, Alice) such that Luke is not the *worst valid partner* of Alice.
- Let Bob be Alice's *worst valid partner*.
- Then there is a stable matching S with (Bob,Alice)
- Contradiction: S is not stable.
  - By assumption, Alice likes Luke better than her partner Bob in S
  - Luke likes Alice better than his partner in S
    - We already know that Alice is his optimal match !

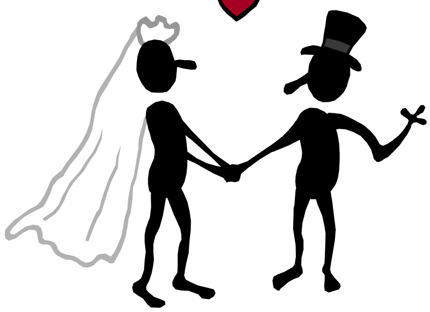
Luke

Alice

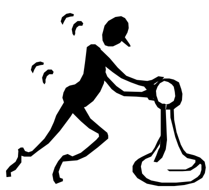
Conclusions

☺

Marry Well!



Advice to females



- Learn to make the first move.

Lessons

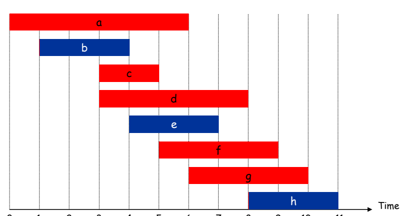
- Isolate / Abstract out structures
- Create Efficient Algorithms
- History
  - Why do men propose?

1.2 Five Representative Problems

Interval Scheduling

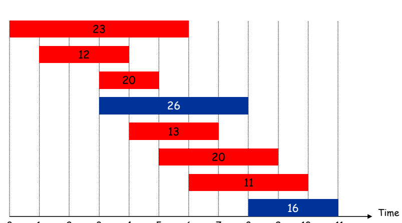
- Input. Set of jobs with start times and finish times.
- Goal. Find maximum cardinality subset of mutually compatible jobs.

jobs don't overlap



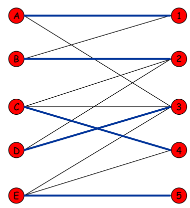
Weighted Interval Scheduling

- Input. Set of jobs with start times, finish times, and weights.
- Goal. Find maximum weight subset of mutually compatible jobs.



Bipartite Matching

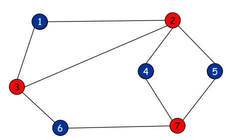
- Input. Bipartite graph.
- Goal. Find maximum cardinality matching.



Independent Set

- Input. Graph.
- Goal. Find maximum cardinality independent set.

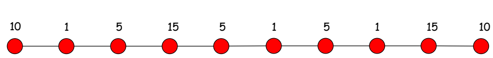
subset of nodes such that no two joined by an edge



68

Competitive Facility Location

- Input. Graph with weight on each each node.
- Game. Two competing players alternate in selecting nodes. Not allowed to select a node if any of its neighbors have been selected.
- Goal. Select a maximum weight subset of nodes.



Second player can guarantee 20, but not 25.

69

Five Representative Problems

- Variations on a theme: independent set.

- Interval scheduling: n log n greedy algorithm.
- Weighted interval scheduling: n log n dynamic programming algorithm.
- Bipartite matching: n^k max-flow based algorithm.
- Independent set: NP-complete.
- Competitive facility location: PSPACE-complete.

70

REFERENCES

•D. Gale and L. S. Shapley, *College admissions and the stability of marriage*, American Mathematical Monthly 69 (1962), 9-15

•Dan Gusfield and Robert W. Irving, *The Stable Marriage Problem: Structures and Algorithms*, MIT Press, 1989