Complexity
CS 320, Fall 2017

Dr. Geri Georg, Instructor
georg@colostate.edu

Some materials adapted from Prof. Wim Bohm
Interval Scheduling

(Greedy algorithms)
Weighted Intervals
(dynamic programming)
A graph is bipartite if its nodes can be partitioned in two sets X and Y, such that the edges go from one x in X to a y in Y.

How is this different from stable matching?
Independent Set
(no known polynomial algorithm)

subset of nodes such that no two are joined by an edge
Tractable – Polynomial time

Brute-force checking every possible solution:

\[ 2^n \text{ time or worse for inputs of size } n \]

Polynomial:

on every input of size \( n \), running time is bounded by \( cn^d \) steps

\( c \) and \( d \) are constants: \( c > 0 \) and \( d > 0 \)

We want to quantify running time in a way that allows us to identify broad classes of algorithms.
Asymptotic Upper Bound

\[ f(n) = O(g(n)) : \]
- constants \( n_0, c > 0 \)
- for all \( n > n_0 \):
- \( f(n) \) is always on or below \( cg(n) \)
Asymptotic Lower Bound

\[ f(n) = \Omega(g(n)) : \]
- constants \( n_0, c > 0 \)
- for all \( n > n_0 \):
- \( f(n) \) is always on or above \( cg(n) \)
Asymptotic Tight Bound

\[ f(n) = \Theta(g(n)) : \]
- constants \( n_0, \)
- \( c_1, c_2 > 0 \)
- for all \( n > n_0 : \)
- \( f(n) \) is always between \( c_1g(n) \) and \( c_2g(n) \)
Implementing Gale-Shapley

1. Every pet and every human start out free.
2. While (some pet \( p \) is free and hasn’t considered every human):
   3. Choose \( p \).
   4. \( h = \) highest-ranked human in \( p \)’s list who hasn’t been considered.
   5. If \( h \) is free, \((p, h)\) are in a trial.
   6. Else if (\( h \) prefers \( p \) to current match \( p' \)), \((p, h)\) enter a trial and \( p' \) becomes free.
   7. Else \( h \) prefers \( p' \) and \( p \) remains free.
8. Return the set \( S \) of matched pets and humans.
Things to Consider

VERSION WHERE PETS CHOOSE

1. every pet and every human start out free

4. $h =$ highest-ranked human in $p$’s list who hasn’t been considered

5. if $h$ is free, $(p, h)$ are in a trial

6. else if ($h$ prefers $p$ to current $p’$), trial is $(p, h)$ and $p’$ becomes free

7. return the set $S$ of matched pets and humans
Array \text{prefs}[n], n = 5 (index is preference, from highest to lowest)

- access element \textit{i} in \textit{O}(1) time
- query if \textit{j} in array takes \textit{O}(n) time
List of free pets?
• dynamically changes linked list:
  • easy to insert, delete at beginning
  • have to traverse to find $i^{th}$ element $O(i)$
Actions in the loop

3. Choose p

4. \( h = \text{highest-ranked human in } p' \text{'s list who hasn't been considered} \)

5. If \( h \) is free, \( (p, h) \) are in a trial

6. Else if (\( h \) prefers \( p \) to current match \( p' \)), \( (p, h) \) enter a trial and \( p' \) becomes free
O(1) Data Structures

Identify free pet
• linked list: insert/delete at front

Identify next human to consider
• array Next[n]: next human j for pet i

Identify if human free and if not, what pet is current match
• array Current[n]: Current[i] is human i’s current match, NULL if not matched yet

Identify which pet is preferred if in match
• Array Ranking[n,n]: Ranking[i, j] is the rank of pet j in the preference list of human i
Next[ ] = 1

PetPrefs[ , 1] =

Current [ ] = NULL

But what if it isn’t NULL? how to find which pet is preferred in O(1) time?
Create rank ordering array for human’s preferences. We’ll use this ordering for pets:

1  2  3  4  5

So, for , the preference order is:

And the ranking array row would look like:
Creating the match is then a simple matter of updating the Current array:

\[
\text{Current }[ ] = \text{ [image]
}

And when the loop terminates we return the matches in Current[]

Python has some better data structures we can use – what are your ideas?
Image Credits


Representative problem images: Prof. Wim Bohm, Computer Science Department, CSU

lotsCompl: https://en.wikipedia.org/wiki/Analysis_of_algorithms

bigTheta: https://www.kullabs.com/classes/subjects/units/lessons/notes/note-detail/3930

bigOmega: https://www.kullabs.com/classes/subjects/units/lessons/notes/note-detail/3930

graph-bigO: https://www2.hawaii.edu/~janst/311/Notes/Topic-03.html

Great Dane: https://www.google.com/search?q=great+dane&tbm=isch&tbs=rimg:CdUOPFELC7ujiX3ZM6iHig6Pz4svwSGnVjLOeivhNfs3iBbw69cMyGhEsqYUNUG_1lajFGVZeGlC6z6IkwpEshyoSCZfdKzqjeKDoEQMUxUbUmJfKhiJIIPiz61BiadURy5uw2Z1acvogEgmOU56K-E1-zRHYV9GE3BHb0ioScERMvDr1wzlafB1qvtMlwkJ4KhjESyphQ1QbURHbLA0fAVSGwqEglqAVUZI4YuXEY80_1PflgVwSocSTCpPqWTCKSwfEfhPzkZNOKkPB&tbo=u&sa=X&ved=0ahUKEwi3tL2IbHVAhVj44MKHXIWDA9C8IHW&biw=1172&bih=799&dpr=1.15#imgrc=y3Yo4qQaxoINM:

Yorkshire Terrier: https://www.pinterest.com/explore/yorkshire-terrier-puppies/


Parakeet: https://www.pinterest.com/explore/parakeet/

Person with a Beard: https://www.pinterest.com/pin/475692779362324515/

Person wearing a Hat: http://imgur.com/gallery/trtuk

People wearing Red shirts: https://neevov.wordpress.com/2014/02/26/buy-red-t-shirts-for-men-and-women-in-india/

People wearing Sandals: https://www.pinterest.com/fabulouscolours/sandal-scandal/