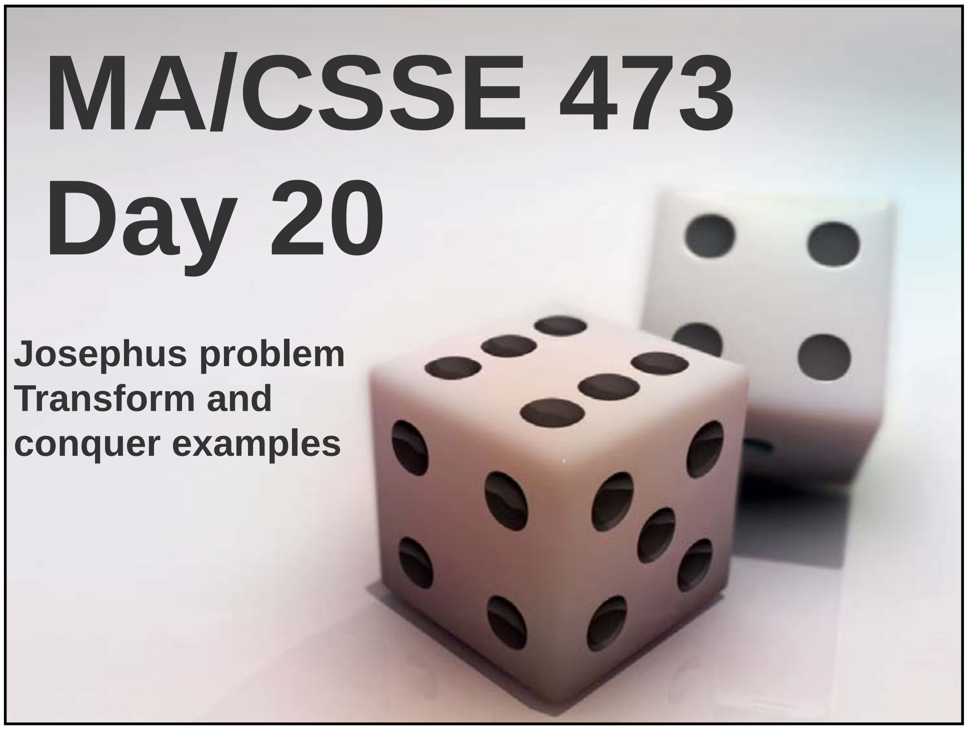
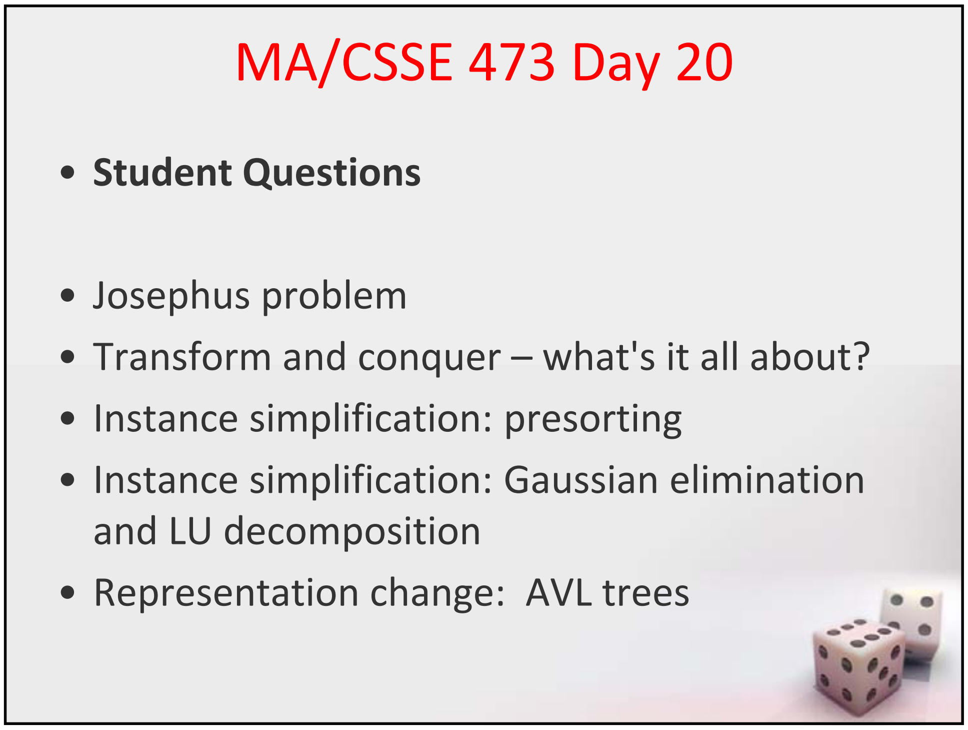


MA/CSSE 473 Day 20

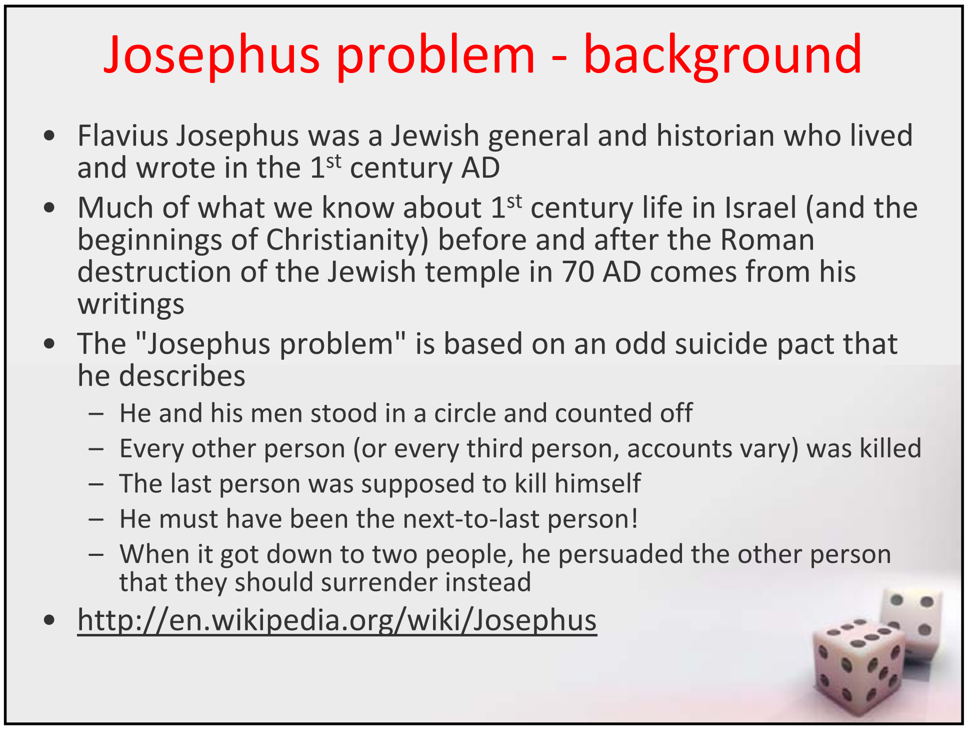
Josephus problem
Transform and conquer examples



MA/CSSE 473 Day 20

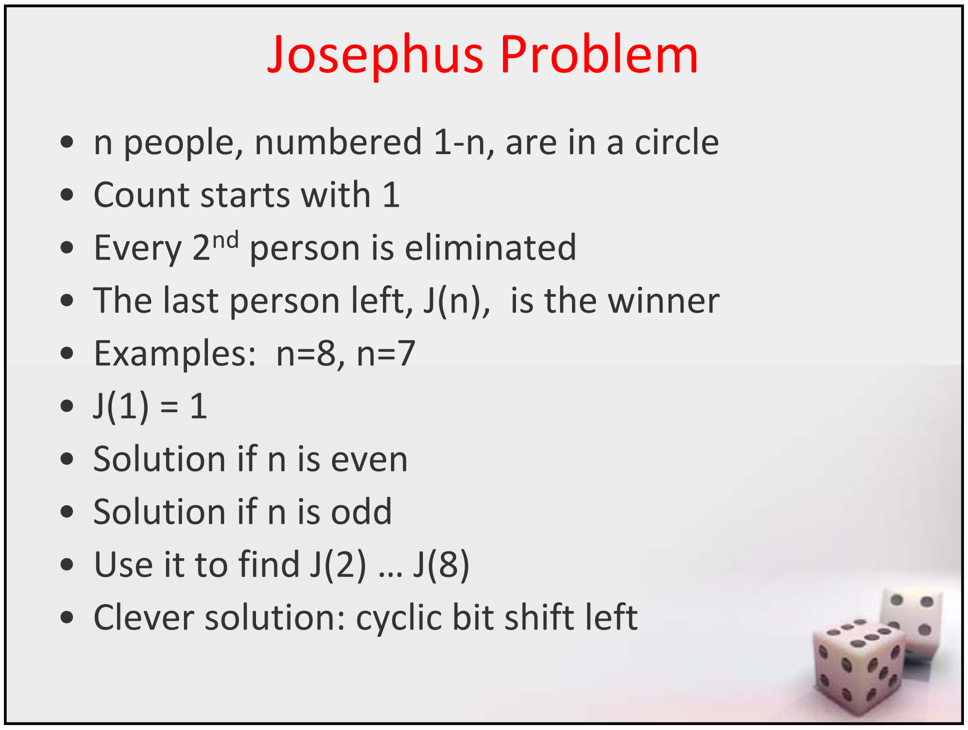
- **Student Questions**

- Josephus problem
- Transform and conquer – what's it all about?
- Instance simplification: presorting
- Instance simplification: Gaussian elimination and LU decomposition
- Representation change: AVL trees



Josephus problem - background

- Flavius Josephus was a Jewish general and historian who lived and wrote in the 1st century AD
- Much of what we know about 1st century life in Israel (and the beginnings of Christianity) before and after the Roman destruction of the Jewish temple in 70 AD comes from his writings
- The "Josephus problem" is based on an odd suicide pact that he describes
  - He and his men stood in a circle and counted off
  - Every other person (or every third person, accounts vary) was killed
  - The last person was supposed to kill himself
  - He must have been the next-to-last person!
  - When it got down to two people, he persuaded the other person that they should surrender instead
- http://en.wikipedia.org/wiki/Josephus

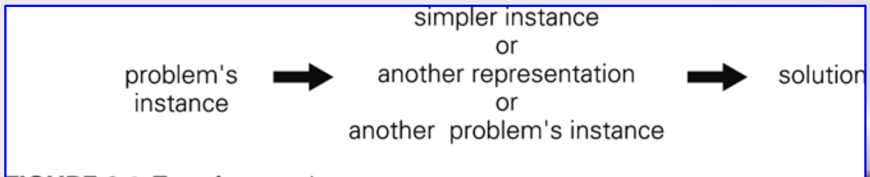


Josephus Problem

- n people, numbered 1-n, are in a circle
- Count starts with 1
- Every 2nd person is eliminated
- The last person left, $J(n)$, is the winner
- Examples: $n=8$, $n=7$
- $J(1) = 1$
- Solution if n is even
- Solution if n is odd
- Use it to find $J(2) \dots J(8)$
- Clever solution: cyclic bit shift left

Transform and Conquer Algorithms

- Transform a problem to a simpler instance of the same problem – **instance simplification**
- Transformation to a different representation of the same instance – **representation change**
- Transformation to an instance of a different problem that we know how to solve – **problem reduction**



problem's instance
simpler instance or another representation or another problem's instance
solution

FIGURE 6.1 Transform-and-conquer strategy

Instance simplification: Presorting an Array

- The following problems are simplified by pre-sorting the array:
  - **Search** (can do Binary or Interpolation search)
  - Determine whether the array contains **duplicates**
  - Find the **median** of the array
  - Find the **mode** of the elements of the array
    - The most frequently-occurring element
  - A related problem: Anagrams
    - In a large collection of words, find words that are anagrams of each other
    - How can pre-sorting help?
    - Sort the letters of each word
  - Interval union problem from early part of PLC

Instance Simplificaiton: Gaussian Elimination (hopefully you saw it in a DE class)

- Solve a system of n linear equations in n unknowns
  - Represent the system by an augmented coefficient matrix
  - Transform the matrix to triangular matrix by a combination of the following solution-preserving elementary operations:
    - exchange two rows
    - multiply a row by a nonzero constant
    - replace a row by that row plus or minus a constant multiple of a different row
  - Look at the algorithm and analysis on pp 207-208; if you can't understand them, ask at some point.
  - Θ(n^3) [previous HW]

Other Applications of G.E.

- Matrix inverse
  - Augment a square matrix by the identity matrix
  - Perform elementary operations until the original matrix is the identity.
  - The "augmented part" will be the inverse
  - More details and an example at http://en.wikipedia.org/wiki/Gauss-Jordan_elimination

Other Applications of G.E.

- Determinant calculation
  - Calculation of the determinant of a triangular matrix is easy
- What effect does each of the elementary operations have on the determinant?
  - exchange two rows
  - multiply a row by a nonzero constant
  - replace a row by that row plus or minus a constant multiple of a different row
- Do these operations until you get a triangular matrix
- Keep track of the operations' cumulative effect on the determinant

LU Decomposition

- This can speed up all three applications of Gaussian Elimination
- Write the matrix A as a product of a Lower Triangular matrix L and an upper Triangular matrix U.
- Example: $[A]=\begin{bmatrix}25 & 5 & 1\\ 64 & 8 & 1\\ 144 & 12 & 1\end{bmatrix}$

$$[L]=\begin{bmatrix}1 & 0 & 0\\ 2.56 & 1 & 0\\ 5.76 & 3.5 & 1\end{bmatrix} \quad [U]=\begin{bmatrix}25 & 5 & 1\\ 0 & -4.8 & -1.56\\ 0 & 0 & 0.7\end{bmatrix}$$

Representation change: AVL Trees (what you should remember…)

- Named for authors of original paper, **A**delson-**V**elskii and **L**andis (1962).
- An AVL tree is a height-balanced Binary Search Tree.
- A BST T is **height balanced** if T is empty, or if
  - $|\text{height}(T_L) - \text{height}(T_R)| \leq 1$, and
  - T_L and T_R are both height-balanced.
- Show: Maximum height of an AVL tree with N nodes is $\Theta(\log N)$.
- How do we maintain balance after insertion?
- **Exercise:** Given a pointer to the root of an AVL tree with N nodes, find the height of the tree in log N time.
- Details on balance codes and various rotations are in the CSSE 230 slides that are linked from the schedule page.