Objectives

- Divide and conquer problems
- Recurrence relations
- Problem: Counting inversions

Review

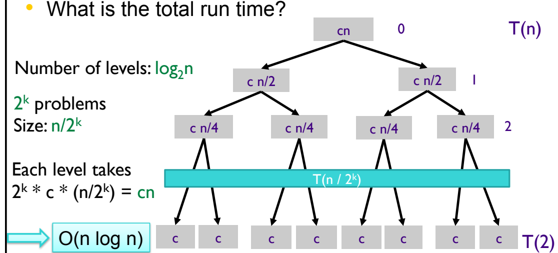
- What is a divide and conquer algorithm?
- Name two ways to determine the runtime for a recurrence relation
  - What is the first step for either approach?
- What is a recurrence relation?

Review: Approaches to Solving Recurrences

1. Unroll recursion
   - Look for patterns in runtime at each level
   - Sum up running times over all levels
2. Substitute guess solution into recurrence
   - Check that it works
   - Induction on n

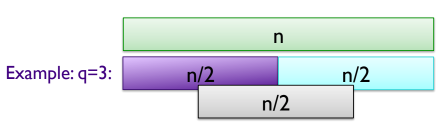
Review: Unrolling Recurrence
T(n) = 2T(n/2) + O(n)

- How many levels are there (assuming *n* is a power of 2)?
- How much does each level cost, in terms of the level?
- What is the total run time?



Another Recurrence Relation

- Instead of recursively solving 2 problems, solve *q* problems
  - Size of problems is still n/2
- Combining solutions is still O(n)



What is the recurrence relation?

Another Recurrence Relation

- Instead of recursively solving 2 problems, solve *q* problems
  - Size of problems is still n/2
- Combining solutions is still O(n)
- Recurrence relation:
  - For some constant *c*,

    $T(n) \le q\,T(n/2) + cn$ when $n > 2$

    $T(2) \le c$

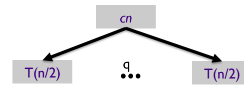
Intuition about running time?

Unrolling Recurrence, q > 2

$T(n) \leq q\, T(n/2) + cn$

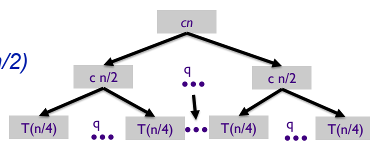
Unrolling Recurrence, q > 2

- First level:
  q T(n/2) + *cn*



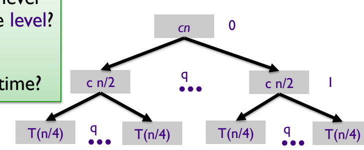
Unrolling Recurrence, q > 2

- Next level:
  q T(n/4) + *c(n/2)*



Unrolling Recurrence, q > 2

How much does each level cost, in terms of the level?
Number of levels?
What is the total run time?



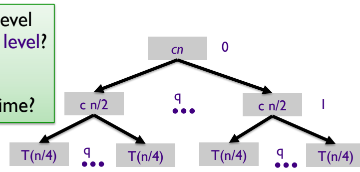
q^k problems at level *k*
Size: $n/2^k$

Number of levels: $\log_2 n$

Each level takes $q^k * c * (n/2^k) = (q/2)^j\, cn$
→ Total work per level is *increasing* as level increases

Unrolling Recurrence, q > 2

How much does each level cost, in terms of the level?
Number of levels?
What is the total run time?



$T(n) \leq \Sigma_{j=0,\log n} (q/2)^j\, cn$

Geometric series:
(constant ratio between successive terms)
Multiplying previous term by (q/2)

$O(n^{\log_2 q})$

Unrolling the Recurrence

- Generalize: What are the steps?

Summary

- Use recurrences to analyze the run time of divide and conquer algorithms
- Need to figure out
  - Number of sub problems
  - Size of sub problems
  - Number of times divided (number of levels)
  - Cost of merging problems

Know Your Recurrence Relations

What algorithm has this recurrence relation?
What is that algorithm's running time?

Recurrence	Algorithm	Running Time
T(n) = T(n/2) + O(1)		
T(n) = T(n-1) + O(1)		
T(n) = 2 T(n/2) + O(1)		
T(n) = T(n-1) + O(n)		
T(n) = 2 T(n/2) + O(n)		

Know Your Recurrence Relations

What algorithm has this recurrence relation?
What is that algorithm's running time?

Recurrence	Algorithm	Running Time
T(n) = T(n/2) + O(1)	Binary Search	O(log n)
T(n) = T(n-1) + O(1)	Sequential/ Linear Search	O(n)
T(n) = 2 T(n/2) + O(1)	Binary Tree Traversal	O(n)
T(n) = T(n-1) + O(n)	Selection Sort	O(n^2)
T(n) = 2 T(n/2) + O(n)	Merge Sort	O(n log n)

COUNTING INVERSIONS

Problem Context

- Movie site tries to match your movie preferences with others
  - You rank *n* movies
  - Movie site consults database to find people with similar tastes
    - **Collaborative filtering**
- Meta-search tools
  - Do same query on several search engines
  - Synthesize results by looking for similarities and differences in search engines' results rankings

Comparing Rankings

- To determine similarity of rankings, need a metric
- Similarity metric: number of inversions between two rankings
  - My rank: 1, 2, …, n
  - Your rank: a_1, a_2, …, a_n
  - Movies i and j *inverted* if i < j but $a_i > a_j$

Discuss pros and cons of this metric

Movies

	A	B	C	D	E
Me	1	2	3	4	5
You	1	3	4	2	5

What are the inversions?

Comparing Rankings

- To determine similarity of rankings, need a metric
- Similarity metric: number of inversions between two rankings
  - My rank: 1, 2, …, n
  - Your rank: a_1, a_2, …, a_n
  - Movies i and j *inverted* if i < j but $a_i > a_j$

Naïve/Brute force solution?

Movies

	A	B	C	D	E
Me	1	2	3	4	5
You	1	3	4	2	5

Inversions:
3-2, 4-2

Brute Force Solution

- Look at every pair (i,j) and determine if they are an inversion
- Requires $\Theta(n^2)$ time
  - Note: Already an efficient algorithm but try to improve upon runtime

Towards a Better Solution…
- Can't look at each inversion individually

Applications

- Voting theory
- Collaborative filtering
- Measuring the "sortedness" of an array
- Sensitivity analysis of Google's ranking function
- Rank aggregation for meta-searching on the Web
- Nonparametric statistics (e.g., Kendall's Tau distance)

Counting Inversions: Divide-and-Conquer

Assume number represents where item *should* be in the list, i.e., where it is in someone else's list

1	5	4	8	10	2	6	9	12	11	3	7

Counting Inversions: Divide-and-Conquer

- Divide: separate list into two pieces

1	5	4	8	10	2	6	9	12	11	3	7

Divide: O(1)

1	5	4	8	10	2

6	9	12	11	3	7

What are some inversions in blue and green halves?

Counting Inversions: Divide-and-Conquer

- Divide: separate list into two pieces
- Conquer: recursively count inversions in each half

1	5	4	8	10	2	6	9	12	11	3	7

Divide: O(1)

1	5	4	8	10	2

6	9	12	11	3	7

What is recurrence relation so far?

5 blue-blue inversions
5-4, 5-2, 4-2, 8-2, 10-2

8 green-green inversions
6-3, 9-3, 9-7, 12-3, 12-7, 12-11, 11-3, 11-7

Counting Inversions: Divide-and-Conquer

- Divide: separate list into two pieces
- Conquer: recursively count inversions in each half

1	5	4	8	10	2	6	9	12	11	3	7

Divide: O(1)

1	5	4	8	10	2	6	9	12	11	3	7

Conquer: 2T(n / 2)

5 blue-blue inversions
5-4, 5-2, 4-2, 8-2, 10-2

8 green-green inversions
6-3, 9-3, 9-7, 12-3, 12-7, 12-11, 11-3, 11-7

What do we need to do next?

Counting Inversions: Divide-and-Conquer

- Divide: separate list into two pieces
- Conquer: recursively count inversions in each half
- Combine: count inversions where a_i and a_j are in different halves, and return sum of three quantities

1	5	4	8	10	2	6	9	12	11	3	7

Divide: O(1)

1	5	4	8	10	2	6	9	12	11	3	7

Conquer: 2T(n / 2)

5 blue-blue inversions

8 green-green inversions

9 blue-green inversions
5-3, 4-3, 8-6, 8-3, 8-7, 10-6, 10-9, 10-3, 10-7

Combine cost: ???

Total = 5 + 8 + 9 = 22

Counting Inversions: Divide-and-Conquer

- Divide: separate list into two pieces
- Conquer: recursively count inversions in each half
- Combine: count inversions where a_i and a_j are in different halves, and return sum of three quantities

1	5	4	8	10	2	6	9	12	11	3	7

Divide: O(1)

1	5	4	8	10	2	6	9	12	11	3	7

Conquer: 2T(n / 2)

5 blue-blue inversions

8 green-green inversions

9 blue-green inversions
5-3, 4-3, 8-6, 8-3, 8-7, 10-6, 10-9, 10-3, 10-7

Combine: seems like $\Theta(n^2)$

Total = 5 + 8 + 9 = 22

What would make figuring out blue-green inversions easier?

Looking Ahead

- Wiki due Wednesday
  - Read 4.7-4.8, 5.1
- PS6 due Friday
- Monday, March 12
  - Katherine Crowley talk at 7:30 p.m. in Stackhouse
  - 5 pts extra credit for write up on Sakai