CS140: Algorithms

Z Sweedyk
Lecture 2

Today

- MergeSort
- Divide and conquer algorithms
- QuickSort
- Average Case Analysis

Merge-sort

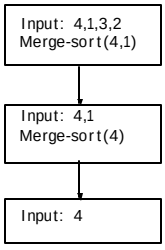
Merge-sort($S=\{s_1, s_2, \dots, s_n\}$)
If n=1 return(S)
Else
S_1 = Merge-sort($s_1, \dots, s_{\lfloor n/2 \rfloor}$)
S_2 = Merge-sort($s_{\lfloor n/2 \rfloor+1}, \dots, s_n$)
Return Merge(S_1, S_2)

Merge($s_1,s_2,\dots,s_k; t_1,t_2,\dots,t_j$)

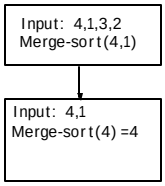
($k>0$ and $j>0$)

- If $s_1 \le t_1$ then output
  s_1, Merge($s_2,\dots,s_k; t_1,t_2,\dots,t_j$)
- Else output
  t_1, Merge($s_1, s_2,\dots,s_k; t_2,\dots,t_j$)

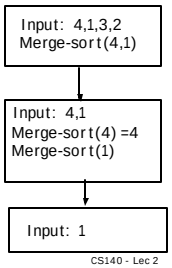
Merge-sort(4,1,3,2)



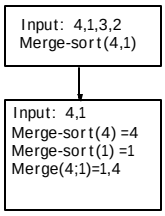
Merge-sort(4,1,3,2)



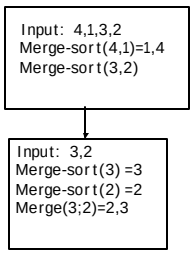
Merge-sort(4,1,3,2)



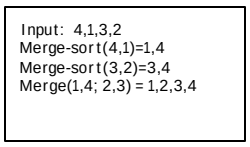
Merge-sort(4,1,3,2)



Merge-sort(4,1,3,2)



Merge-sort(4,1,3,2)



Is Merge-sort correct?

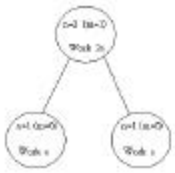
- If n=1 then yes
- If n>1 then
  - We can assume Merge-sort(S(1),…,S($\lfloor n/2 \rfloor$)) and Merge-sort(S($\lfloor n/2 \rfloor$ +1), … , S(n)) return correctly sorted lists.
  - So the merge of these lists is a correctly sorted list.

How fast is Merge-sort? (Assume $n=2^m$)

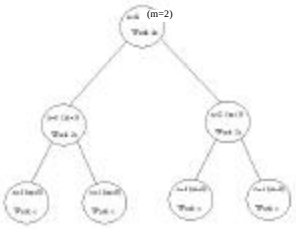
- m=0: $T(1) = c$
- m>0: $T(2^m) = 2T(2^{m-1}) + c2^m$

Work Tree for Merge-sort
Input Size: 1 (m=0)

Work Tree for Merge-sort
Input Size: 2 (m=1)



Work Tree for Merge-sort
Input Size: 4 (m=2)



Work Tree for Merge-sort
Input Size: $n=2^m$

A root with two sub-trees
- Root
  - Input Size: n
  - Work: cn
- Each child
  - Roots a work tree with Input Size 2^{m-1}

Work Tree for Merge-sort
Input Size: $n=2^m$

Properties of nodes at level i (root is at level 0):
- Input size:
- Work:

Properties of level i:
- Number of nodes at level i:
- Total work of nodes at level i:

Property of tree:
- Number of levels:
- Total work:

Work Tree for Merge-sort
Input Size: $n=2^m$

Properties of nodes at level i (root is at level 0):
- Input size: 2^{m-i}
- Work: $c2^{m-i}$

Properties of level i:
- Number of nodes at level i: 2^i
- Total work of nodes at level i: $c2^m$

Property of tree:
- Number of levels: m+1
- Total work: $c(m+1)2^m$ or $O(n \lg n)$

What if $n \neq \lceil\lceil n \rceil\rceil$?

- Claim 1:
  $T(n) = O(T(\lceil\lceil n \rceil\rceil))$
- Claim 2:
  $T(\lceil\lceil n \rceil\rceil) = O(\lceil\lceil n \rceil\rceil \lg \lceil\lceil n \rceil\rceil)$
- Claim 3:
  $\lceil\lceil n \rceil\rceil \lg \lceil\lceil n \rceil\rceil = O(n \lg n)$
- Hence $T(n) = O(n \lg n)$

Today

- MergeSort
- Divide and conquer algorithms
- QuickSort
- Average Case Analysis

Divide and Conquer

"Divide and conquer" is an algorithmic technique:

- Break the problems into ***a*** sub-problems of size ***n/b***
- Solve the sub-problems
- Combine the solutions to the sub-problems to create a solution for the original problem

Divide and Conquer

- "Divide and conquer" recurrence relations

  T(n)=a T(n/b) + f(n)

  T(1)=c=f(1)

Analysis

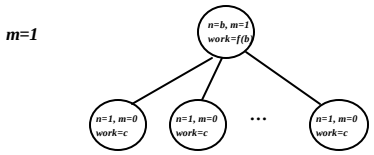
- Consider the case when $n=b^m$
- Then generalize to arbitrary n

Work Tree for Divide and Conquer: $n=b^m$

m=0

Total work: c

Work Tree for Divide and Conquer: $n=b^m$



Total work: $f(b) + ac$

Work Tree for Divide and Conquer: $n=b^m$

A root with a sub-trees
- Root
  - Input Size: $n=b^m$
  - Work: $f(n)$
- Each child
  - Roots a work tree with Input Size b^{m-1}

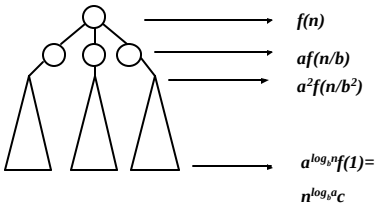
Work Tree for Divide and Conquer

Properties of nodes at level i (root is at level 0):
- Input size:
- Work:

Properties of level i:
- Number of nodes at level i:
- Total work of nodes at level i:

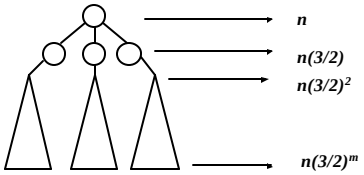
Property of tree:
- Number of levels:
- Total work:

Work Tree for divide and conquer algorithm



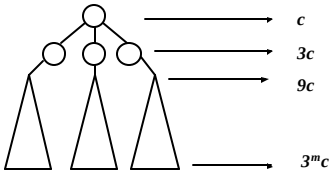
Total work: $\Sigma_{i=0\ldots m} a^i f(n/b^i)$

Work Tree for divide and conquer algorithm: a=3, b=2, f(n)=n



Total work: $\Sigma_{i=0\ldots m} n(3/2)^i$

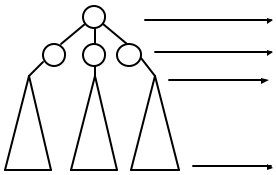
Work Tree for divide and conquer algorithm: a=3, b=2, f(n)=c



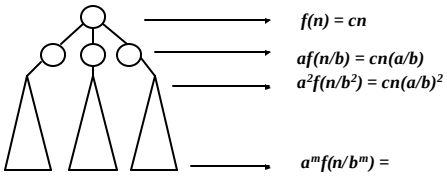
Total work: $\Sigma_{i=0\ldots m} c\,3^i$

Where is "most" of the work?

f(n) is slow-growing ↔ f(n) is fast-growing



$f(n)=cn$, $m=\log_b n$



$T(n) = cn \sum_{i=0\ldots m} (a/b)^i$

Total Work

$T(n) = cn \sum_{i=0\ldots m} (a/b)^i$

$a<b$:

$a=b$:

$a>b$:

Total Work

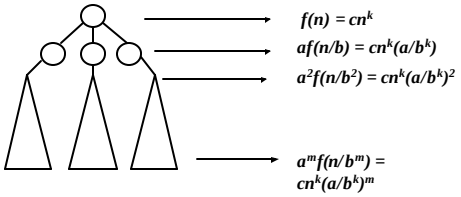
$T(n) = cn \sum_{i=0\ldots m} (a/b)^i$

$a<b$: $O(n)$

$a=b$: $O(n \lg(n))$

$a>b$: $O(n^{\log_b a})$

$f(n)=cn^k$



$T(n) = cn^k \sum_{i=0\ldots m} (a/b^k)^i$

Total Work

$T(n) = cn^k \sum_{i=0\ldots m} (a/b^k)^i$

$a<b^k$:

$a=b^k$:

$a>b^k$:

Total Work

$T(n) = cn^k \sum_{i=0 \ldots m} (a/b^k)^i$

$a<b^k$: $\Theta(n^k)$

$a=b^k$: $\Theta(n^k \lg(n))$

$a>b^k$: $\Theta(n^{\log_b a})$

Today

- MergeSort
- Divide and conquer algorithms
- QuickSort
- Average Case Analysis

Quick-sort

(Assume elements of S are distinct)

Quick-sort(S)

If ||S||=1 return S

Choose pivot x from S

Create

$S_1=\{y \in S \mid y < x\}$

$S_2=\{y \in S \mid y > x\}$

Return (Quick-sort(S_1), x, Quick-sort(S_2))

Quick-sort

- Is Quick-sort correct?
- Is Quick-sort fast?

Is Quick-sort correct?

- If n=1 then yes
- If n>1 then
  - We can assume Quick-sort(S_1) and Quick-sort(S_2) return correctly sorted lists.
  - So Quick-sort(S) is a correctly sorted list.

Is Quick-sort fast?

- $T(1) = c$
- $T(n) = T(n_1) + T(n_2) + cn$

What is worst case?

Is Quick-sort fast?

- $T(1) = c$
- $T(n) = T(n_1) + T(n_2) + cn$

What is worst case?

$T(n) = T(n-1) + cn$

$T(n) = \Theta(n^2)$

Today

- MergeSort
- Divide and conquer algorithms
- QuickSort
- Average Case Analysis

Average-case analysis

What does average-case mean?

- Randomized algorithm on worst-case input
- Deterministic algorithm with a known input distribution
- Deterministic algorithm on worst-case input using amortized cost

A brief tour of (discrete) probability theory…

- Sample space, events, probability
- Discrete probability distributions
- Discrete random variables
- Expectation
- Conditional Probability/Expectation

Experiment 1

- Experiment: A fair coin is flipped
- Sample space:_____
- Events:_____
- Probabilities:_____

Experiment 2

- Experiment: Two fair coins are flipped
- Sample space: _____
- Events:_____
- Probabilities:_____
  _

Experiment 3

- Experiment: A fair die is tossed
- Sample space: _____
- Events: _____
- Probabilities:_____

Discrete Probability Distribution

Assigns a real number to every subset of the sample space such that:

- $P(A) \geq 0$ for any event A
- $\Sigma_{A \in S} P(A) = 1$
- P(A or B) = P(A)+P(B) for disjoint events A,B

Discrete Random Variable X

- Assigns a real number to each outcome.
- Experiment: Toss fair coin
  - If head then X=1
  - If tail then X=0
- Sample space = _____
- Probabilities:
  - What is P(X=0)?
  - What is P(X=1)?
  - What is $P(X \geq 1)$?

Expectation

- $E[X] = \sum_{x \varepsilon S} x\ P(X=x)$
- $E[X^2] = \sum_{x \varepsilon S} x^2\ P(X=x)$
- $Var(X) = E[(X-E[X])^2]$

Discrete Random Variable X

Example continued:
P(0)=P(1)=1/2

Expectation:

$E[X] = \sum_{x \varepsilon S} x$ P(X=x)=_____

$E[X^2] = \sum_{x \varepsilon S} x^2$ P(X=x)=_____

$Var(X) = E[(X-E[X])^2]$=_____

Example

- A biased coin is tossed n times:
  P(H)=p, P(T)=1-p=q
- X is the number of heads
- P(X=k) =_____
  (Binomial distribution)
- E[X] = _____
- $E[X^2]$ = _____

Conditional Probability

- Let A and B be events such that P(B)>0
- Then $P(A|B) = P(A \cap B)/P(B)$

Conditional Probability: Example

Experiment: Toss a fair die

- A is the event that a 2 is rolled
- B is the event the number rolled is even
- C is the event the number rolled is odd
- What is P(A|B)?
- What is P(B|A)?
- What is P(A|C)?
- What is P(C|A)?

Conditional Probability: Properties

Let $A_1, A_2, \ldots, A_k$ be disjoint events that partition the sample space.

Then for any event A

$P(A) = \Sigma_{i=1..k} P(A \mid A_i) P(A_i)$

Average-case analysis: T(n)

What does average-case mean?

- Randomized algorithm on worst-case input
  - $T(n) = Max_{\text{inputs I of size n}} E(\#\text{ steps on input I})$

Average-case analysis: T(n)

What does average-case mean?

- Deterministic algorithm with a known input distribution
  - E[# steps on input I], where I is chosen at random from all inputs of size n

Average-case analysis: T(n)

What does average-case mean?

- Deterministic algorithm on worst-case input using amortized cost
  - We'll talk about this later on

Average-case analysis

What does average-case mean?

- **Randomized algorithm on worst-case input**
- Deterministic algorithm with a known input distribution
- Deterministic algorithm on worst-case input using amortized cost

Randomized Quick-sort

(Assume elements of S are distinct)

Quick-sort(S)

If ||S||=1 return S

Choose pivot x randomly from S

Create

$S_1=\{y \in S \mid y < x\}$

$S_2=\{y \in S \mid y > x\}$

Return (Quick-sort(S_1), x, Quick-sort(S_2))

Analysis of Randomized Quick-Sort

- $T(1) = c$
- $T(n) = T(n_1) + T(n_2) + cn$
- $E[T(n)] = E[T(n_1) + T(n_2) + cn]$
- $E[T(n)] = E[T(n_1) + T(n_2)] + cn$

random

Analysis of Randomized Quick-Sort

$E[T(n_1)+ T(n_2)]$

$= \Sigma_{i=0\ldots n-1} E[T(n_1) + T(n-1-n_1)]$

$= \Sigma_{i=0\ldots n-1} E[T(n_1) + T(n-1-n_1) \mid n_1=i] P(n_1=i)$

$= \Sigma_{i=0\ldots n-1} E[T(i) + T(n-1-i)] P(n_1=i)$

What is $P(n_1=i)$?

Analysis of Randomized Quick-Sort

$E[T(n_1)+ T(n_2)]$

$= \Sigma_{i=0\ldots n-1} E[T(n_1) + T(n-1-n_1)]$

$= \Sigma_{i=0\ldots n-1} E[T(n_1) + T(n-1-n_1) \mid n_1=i] P(n_1=i)$

$= \Sigma_{i=0\ldots n-1} E[T(i) + T(n-1-i)] P(n_1=i)$

$= (1/n) \Sigma_{i=0\ldots n-1} E[T(i) + T(n-1-i)]$

$\leq (1/n) \Sigma_{i=0\ldots n-1} E[2T(\max(i, n-1-i)]$

$= 2/n \Sigma_{i=\lfloor n/2 \rfloor \ldots n-1} E[T(i)]$

Recurrence

- $f(1) = c$
- $f(n) \leq 2/n (\Sigma_{i=\lfloor n/2 \rfloor \ldots n-1} f(i)) + cn$

Guess and prove: $f(n) \leq a\, n \lg n + b$ for some constants a and b

Recurrence

- $f(1) = c$
- $f(n) \leq 2/n \, (\Sigma_{i=\lfloor n/2 \rfloor \ldots n-1} f(i)) + cn$

Base case: $f(n) \leq a \, n \lg n + b$ for $n=1$ provided $c \leq b$

Recurrence

Assume there exists constants a and b such that:

$f(n-1) \leq a \, (n-1) \lg (n-1) + b$

Then

$f(n) \leq 2/n \, (\Sigma_{i=\lfloor n/2 \rfloor \ldots n-1} f(i)) + cn$

$\leq 2/n \, (\Sigma_{i=\lfloor n/2 \rfloor \ldots n-1} (a \, i \lg(i) + b)) + cn$

$\leq 2/n \, (\Sigma_{i=\lfloor n/2 \rfloor \ldots n-1} (a \, i \lg(n) + b)) + cn$

$\leq 2/n \, (\Sigma_{i=1 \ldots n-1} (a \, i \lg(n) + b) - \Sigma_{i=1 \ldots \lfloor n/2 \rfloor - 1} (a \, i \lg(n) + b)) + cn$

and a little algebra later ... ta da