Data Structures and Algorithms

Recurrence Relations Cont.

Olga Karpenko

Parts of this presentation are based on the slides of Prof. David Galles

Announcements

- Homework 2 is out
- Practice session on recursion today at 6:30pm

Recap: Power function

- Power:

```
public int power(int x, int n) {
   if (n == 0)
      return 1;
   else
       return x*power(x, n-1);
}
```

$T(0) = c_1$

$T(n) = c + T(n-1)$ ← Recurrence Relation

Repeated Substitution

$T(n) = T(n-1) + c$ (*)

$T(0) = c_1$

$T(n-1) = T(n-2) + c$, then we can substitute back in the original equation (*):

$T(n) = T(n-2) + c + c = T(n-2) + 2c$

$T(n-2) = T(n-3) + c$, substitute back in:

$T(n) = T(n-3) + c + 2c = T(n-3) + 3c$

$T(n) = T(n-4) + 4c$ ...

Slide is courtesy of Kurt Schmidt

Pattern

- An expression for the kth unwinding:

  $T(n) = T(n-k) + kc$

- Stop at T(0) (empty list)

  $n-k = 0$, when $k = n$

  $T(n) = T(n-n) + nc = T(0) + nc = c_1 + nc$

  no longer have T on the right hand side!

  $T(n) = \text{Theta}(n)$, linear function

Slide is courtesy of Kurt Schmidt

Building a better power function

```
int power(int x, int n) {
      if (n==0) return 1;
      if (n==1) return x;
      if (n % 2 == 0)
        return power(x*x, n/2);
      else
        return power(x*x, n/2) * x;
}
```

Slide is courtesy of Prof. Galles

Building a better power function

```
int power(int x, int n) {
      if (n==0) return 1;
      if (n==1) return x;
      if (n % 2 == 0)
        return power(x*x, n/2);
      else
        return power(x*x, n/2) * x;
}
```

$T(0) = c_1$

$T(1) = c_2$

$T(n) = T(n/2) + c_3$, Same as binary search

Theta(log n)

Exercise: Repeated Substitution

- $T(2) = c$
- $T(n) = T(n^{1/2}) + 1$

Exercise: Repeated Substitution

- $T(2) = c$
- $T(n) = T(n^{1/2}) + 1$

- Answer: T(n) = Theta(log log n)

Solving Recurrence Relations

- Guessing bounds:
  - Repeated Substitution (iteration method)
  - Recursion trees
- Proving bounds: The substitution method
- The Master method (based on Master Theorem)

Recursion Trees

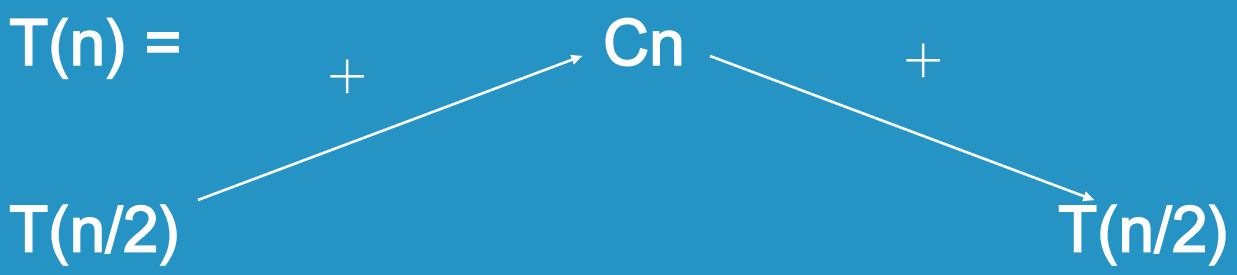
- Another method for solving recurrence relations
- "Visual repeated substitution"

Recursion Trees

- $T(n) = 2\,T(n/2) + Cn$
- $T(1) = C_2$
- $T(0) = C_2$

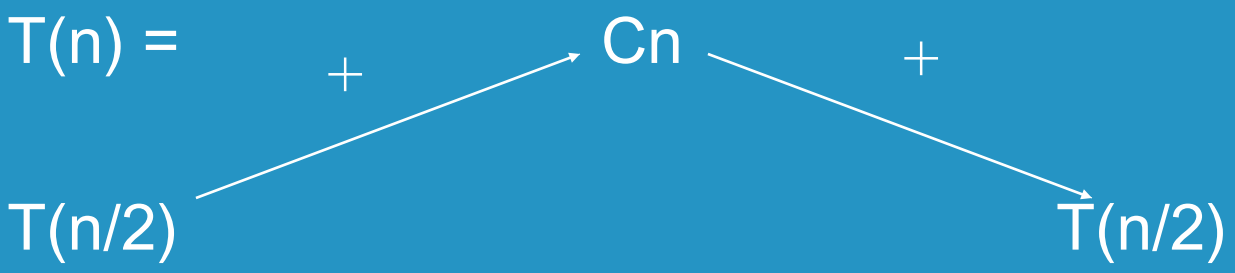
Recursion Trees

- T(n) = 2 T(n/2) + Cn. Rearrange the equation:



Recursion Trees

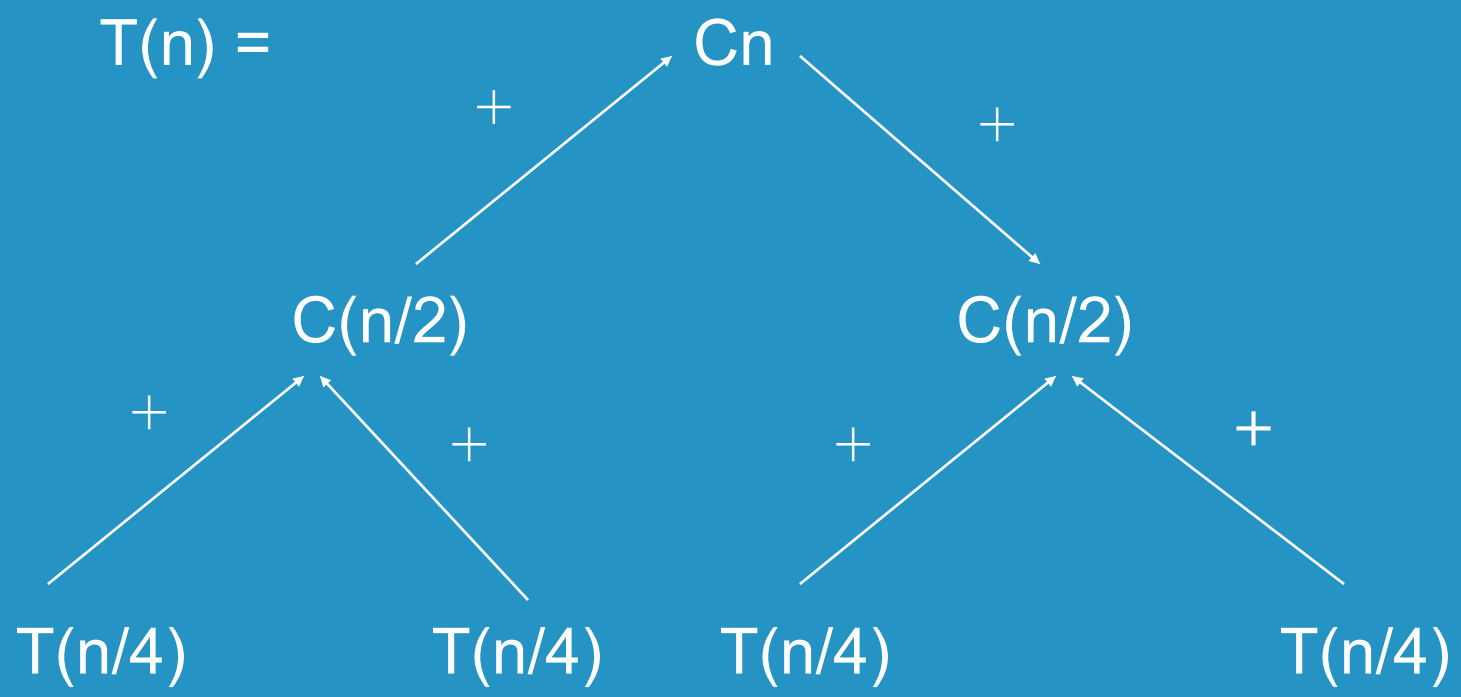
- T(n) = 2 T(n/2) + Cn. Rearrange the equation:



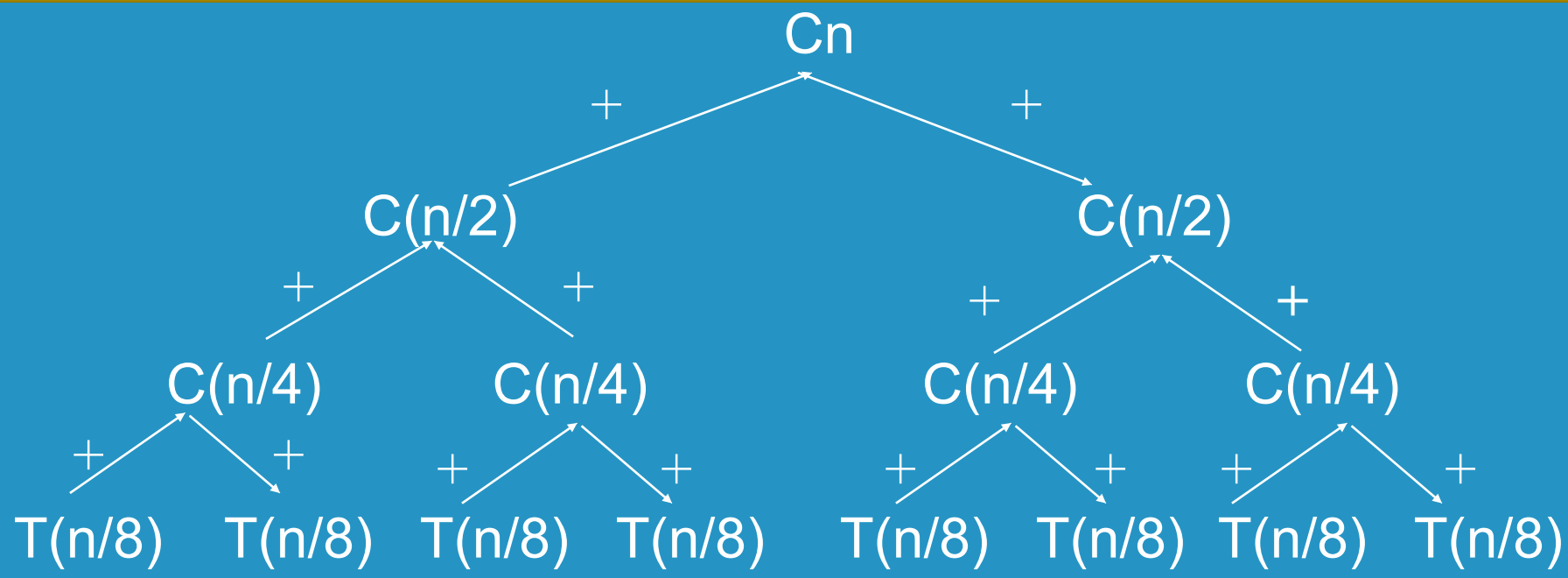
- Since T(n/2) = T(n/4) + T(n/4) + C(n/2), replace T(n/2) with what is on the right side

Recursion Trees

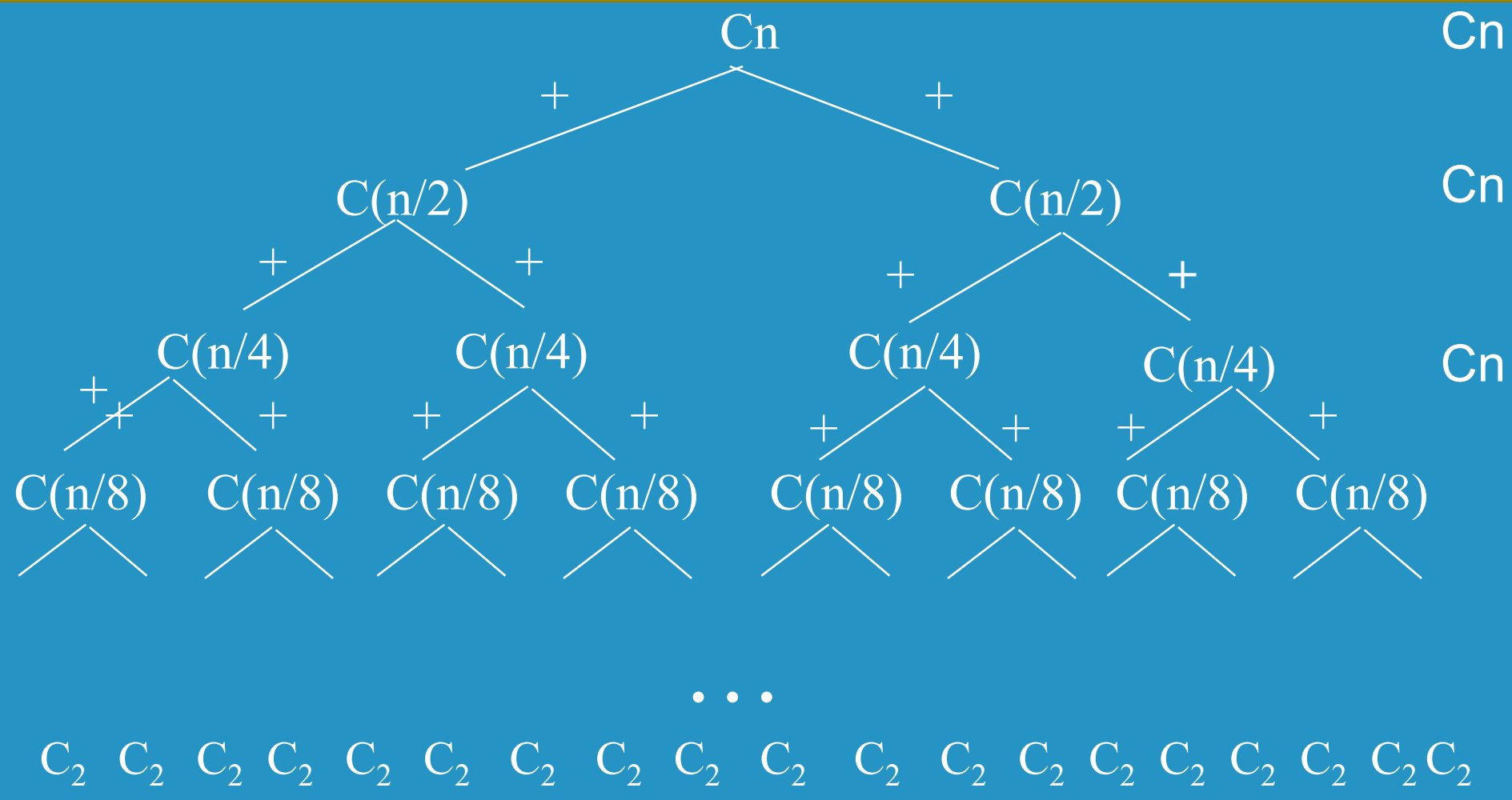
- T(n) = 2 T(n/2) + Cn. Rearrange the equation:



Recursion Trees



Recursion Trees



$2^{\log n} = n$ leaves

Recursion Trees

- Summing it all up: n leaves, the height of the tree is log n:

C_2*n + (log n – 1) Cn = Theta(n log n)

Exercise

- Draw a recursion tree for:

  $T(1) = C_1$

  $T(n) = T(n-1) + c_1$