بسم الله الرحمن الرحیم

ساختمان‌های داده

جلسه ۵

مجتبی خلیلی
دانشکده برق و کامپیوتر
دانشگاه صنعتی اصفهان

مثال

IUT-ECE

```
int example1(int n) {
  int x;
  int y = n - 1;
  x = n + 1;
  y = y + (x * 10)
  int z = x - y;
  return z;
}
```

مثال


```
int example_new1 (int n) {
  int x = 0;
  for (int j = 0; j < 100000; j++) {
    x += i;
  }
  return 0;
}
```

مثال


```
int example2(int n) {
    int s = 0;
    int i = 0;
    while (i < n) {
        s = s + (i * 5);
        i = i + 1;
    }
    return s;
}
```

مثال

IUT-ECE

```
int example3(int n) {
    int s = 0;
    int i = 0;
    while (i < n) {
        int j = 0;
        while (j < n) {
            if (j % 2 == 0) {
                // nothing to do
            }
            s = s + (i * 3) + j;
            j = j + 1;
        }
        i = i + 1;
    } return s;
}
```

مثال


```
int example4(int n) {
   int sum = 0;
   for (int i = n; i > 1; i /= 2)
        sum += i;
   return sum;
}
```

مثال


```
int example_new2(int n) {
  int x = 0;
  for (int i = 0; i < n; i++) {
    for (int j = 0; j < n; j++) {
      x = x + i;
    }
  }
  x += x;
  for (int i = 0; i < n; i++) {
    x = x + i;
  }
  return x;
}
```

مثال

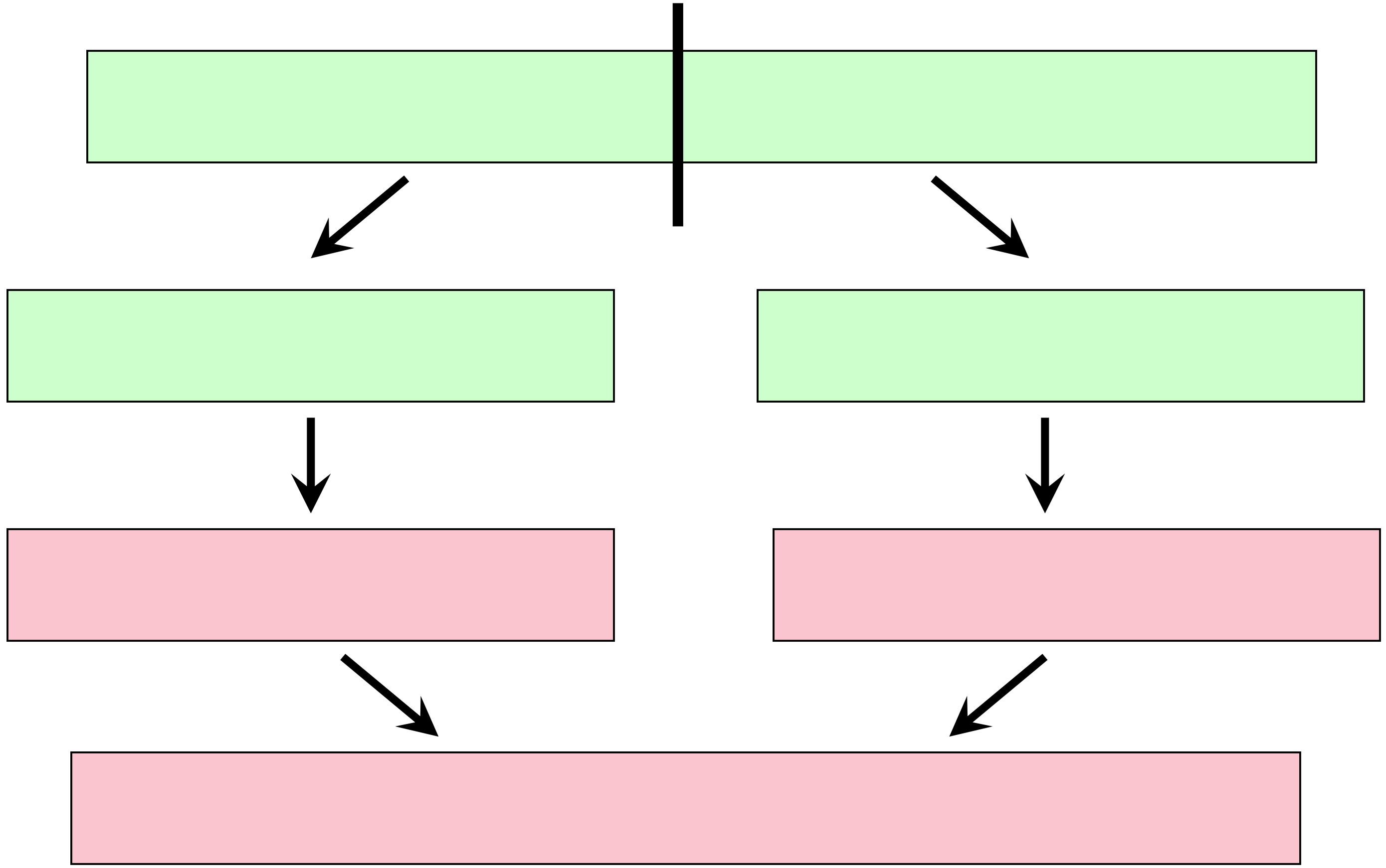

```
int example_new3(int n)
{
  int x = 0;
  if (n % 2 == 0) {
    for (int i = 0; i < n; i++) {
      x += i;
    }
    return x;
  } else {
    return 0;
  }
}
```

یک الگوریتم مرتب‌سازی دیگر

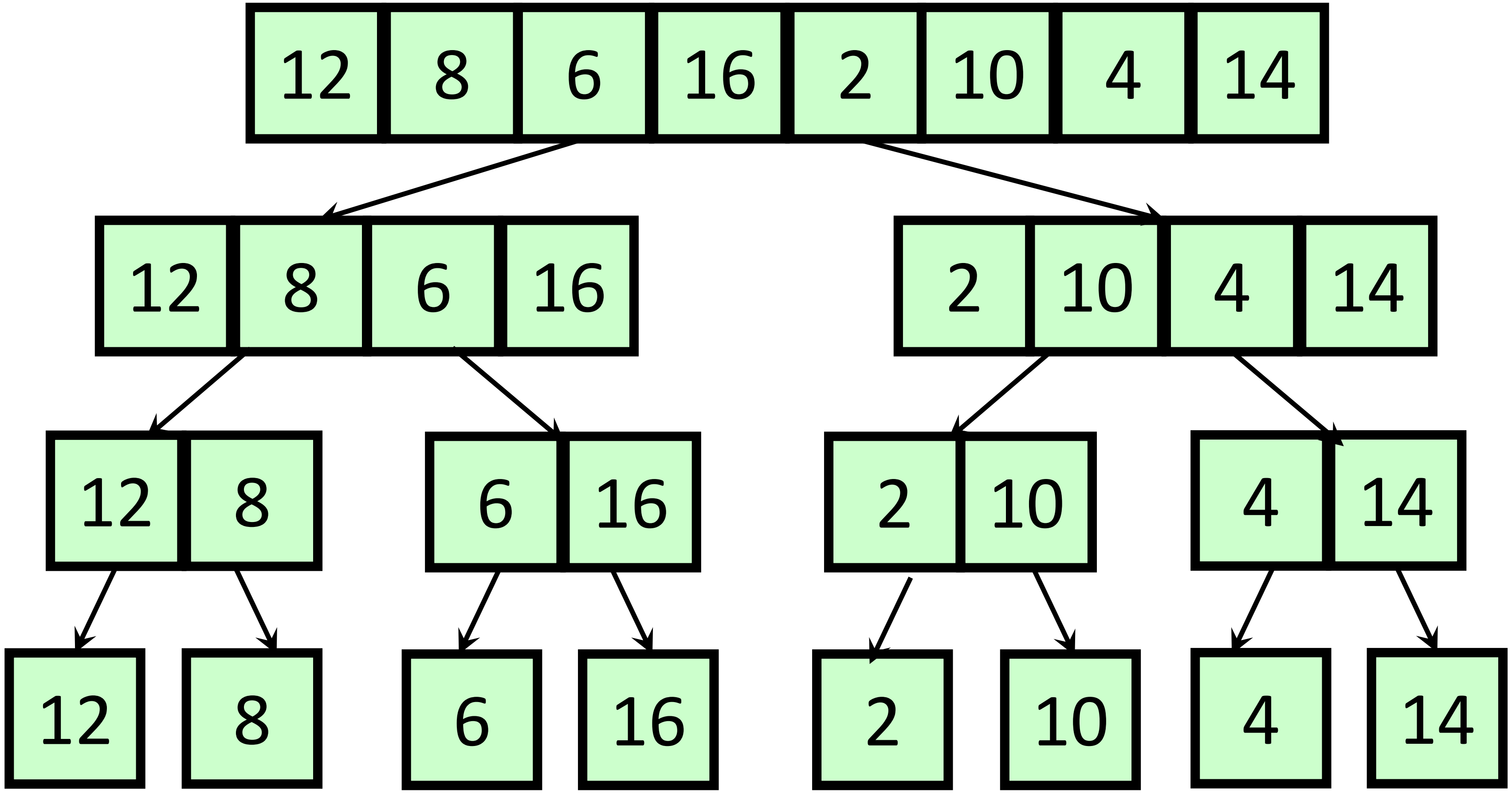
- مرتب‌سازی ادغامی
  - استفاده از تکنیک تقسیم و غلبه برای حل مسئله که به یک الگوریتم بازگشتی منتهی می‌شود.

مرتب‌سازی ادغامی

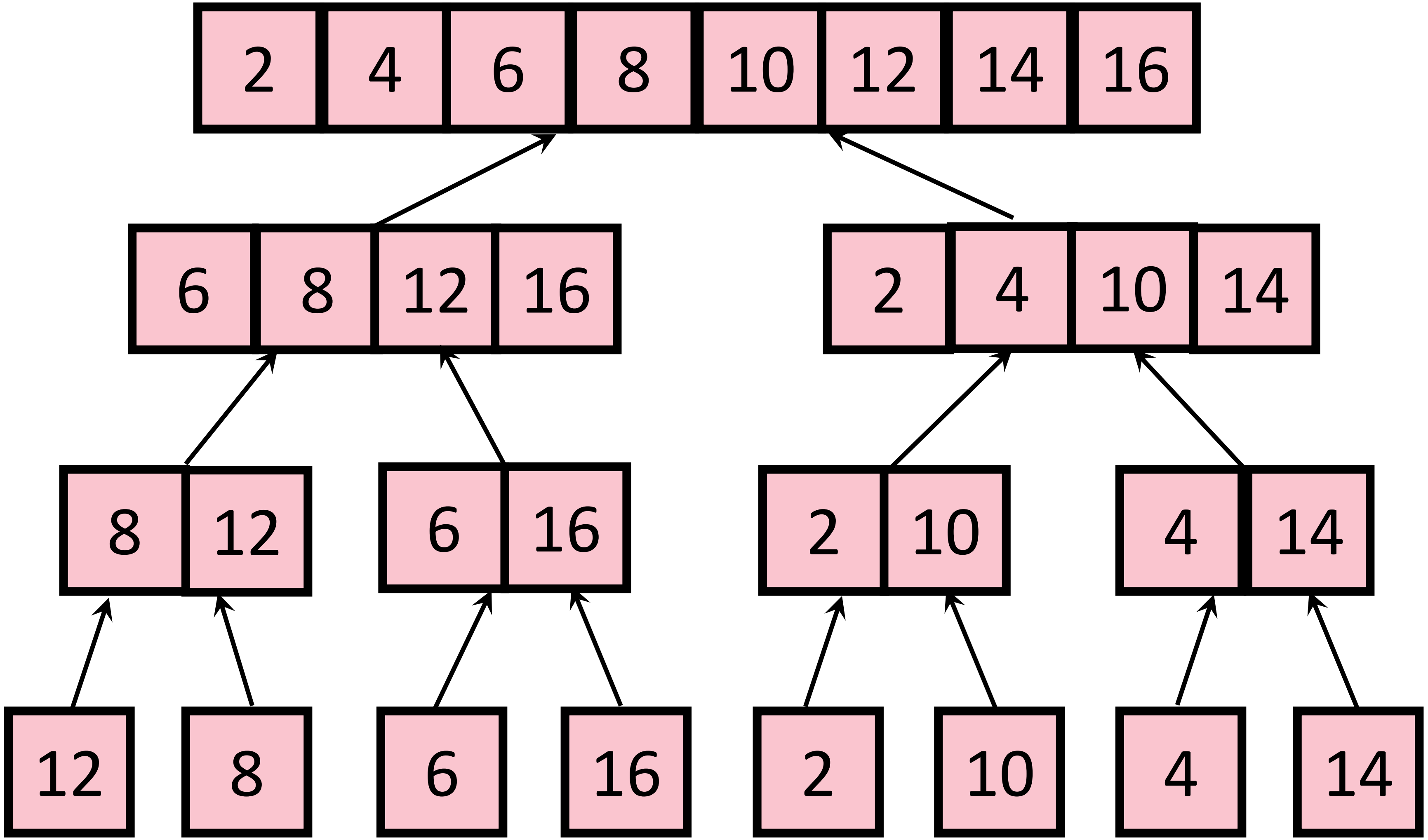
IUT-ECE



مرتب‌سازی ادغامی



مرتب‌سازی ادغامی



مرتب‌سازی ادغامی

IUT-ECE

```
MergeSort(A, l, r) {
   if (l >= r)
      return;
   else
      mid= ⌊l+r(/2⌋;
      leftHalf = MergeSort(A, l, mid);
      rightHalf = MergeSort(A, mid+1, r);
```

مرتب‌سازی ادغامی

IUT-ECE

```
MergeSort(A, l, r) {
   if (l >= r)
      return;
   else
      mid= ⌊(l+r/2)⌋;
      leftHalf = MergeSort(A, l, mid);
      rightHalf = MergeSort(A, mid+1, r);
      return Merge(leftHalf, rightHalf);
}
```

مرتب‌سازی ادغامی

18	21
12	11
5	8
1	3

1 3 5 8 11 12 18 21

مرتب‌سازی ادغامی


```
Merge ( Left, Right ) {
      nl=len(Left); nr=len(Right);  na=nl+nr;
      init (A,na);
      idl=0;  idr=0;

      for (i = 0; i < na; i++){
            if (Left[idl] > Right[idr] ) or (idl >= nl)
                   A[i]= Right[idr];
                   idr++;
            else
                  A[i]= Left[idl];
                  idl++;
      }
      return A;
   }
```

مرتب‌سازی ادغامی

- آیا بدرستی کار می‌کند؟
  - استقرا

مرتب‌سازی ادغامی

- پایدار؟
- وفقی؟
- درجا؟

تحلیل پیچیدگی زمانی


```
MergeSort(A, l, r) {
   if (l >= r)
      return;
   else
      mid= [(l+r)/2];
      leftHalf = MergeSort(A, l, mid);
      rightHalf = MergeSort(A, mid+1, r);
      return Merge(leftHalf, rightHalf);
}
```

تحلیل پیچیدگی زمانی

- در نظر گرفتن بدترین حالت برای الگوریتم Merge

$$T_{merge} = O(nl + nr)$$

تحلیل پیچیدگی زمانی

- پیچیدگی زمانی برای الگوریتم مرتب‌سازی ادغامی (با فرض $n = 2^k$)

$$T(n) = T\left(\frac{n}{2}\right) + T\left(\frac{n}{2}\right) + cn = 2T\left(\frac{n}{2}\right) + cn$$

$$T(n) = \begin{cases} 1 & \text{if } n = 1 \\ 2T(\frac{n}{2}) + cn & \text{otherwise} \end{cases}$$

الگوریتم بازگشتی

IUT-ECE

- یک الگوریتم بازگشتی مسئله را به یک یا چند زیرمسئله کوچکتر تقسیم می‌کند و سپس زیرمسئله‌های کوچکتر را به صورت بازگشتی و با همان الگوریتم حل می‌کند.
- زمان اجرای الگوریتم، حاصل جمع زمان حل زیرمسئله‌ها به اضافه زمان لازم برای تقسیم مسئله به زیرمسئله‌ها و نیز ترکیب جواب‌های به دست آمده است.

راه‌حل‌های بازگشتی

IUT-ECE

- تکنیکی برای طراحی الگوریتم
  - فرض وجود یک راه حل برای مسئله با ورودی کوچکتر
  - استفاده از این راه حل برای همان مسئله با ورودی بزرگتر

حل رابطه بازگشتی

- چگونه به یک فرم بسته برای زمان اجرای الگوریتم مرتب‌سازی ادغامی برسیم؟

$$T(n) = \begin{cases} 1 & \text{if } n = 1 \\ 2T(\frac{n}{2}) + cn & \text{otherwise} \end{cases}$$

حل رابطه بازگشتی

- رابطه‌های بازگشتی را می‌توان به روش‌های زیر حل کرد:
  - حدس و استقراء
  - بسط دادن
  - درخت بازگشت
  - قضیه اصلی

حل رابطه بازگشتی

- به کمک بسط دادن:

$$T(n) = 2T\left(\frac{n}{2}\right) + cn$$
$$= 2\left(2T\left(\frac{n}{4}\right) + \frac{cn}{2}\right) + cn = 4T\left(\frac{n}{4}\right) + 2cn$$
$$= 4\left(2T\left(\frac{n}{8}\right) + \frac{cn}{4}\right) + 2cn = 8T\left(\frac{n}{8}\right) + 3cn$$
$$\dots \qquad = 2^k T\left(\frac{n}{2^k}\right) + kcn$$

$$\text{Let } \frac{n}{2^k} = 1 \Rightarrow 2^k = n \Rightarrow k = \log_2 n$$
$$\Rightarrow T(n) = 2^k T\left(\frac{n}{2^k}\right) + kcn = n\,T(1) + cn\log_2 n$$
$$= \mathcal{O}(n \lg n)$$