ساختمان داده و الگوریتم ها (CE203)

جلسه بیست و پنجم:
نمونه سوال

سجاد شیرعلی شهرضا

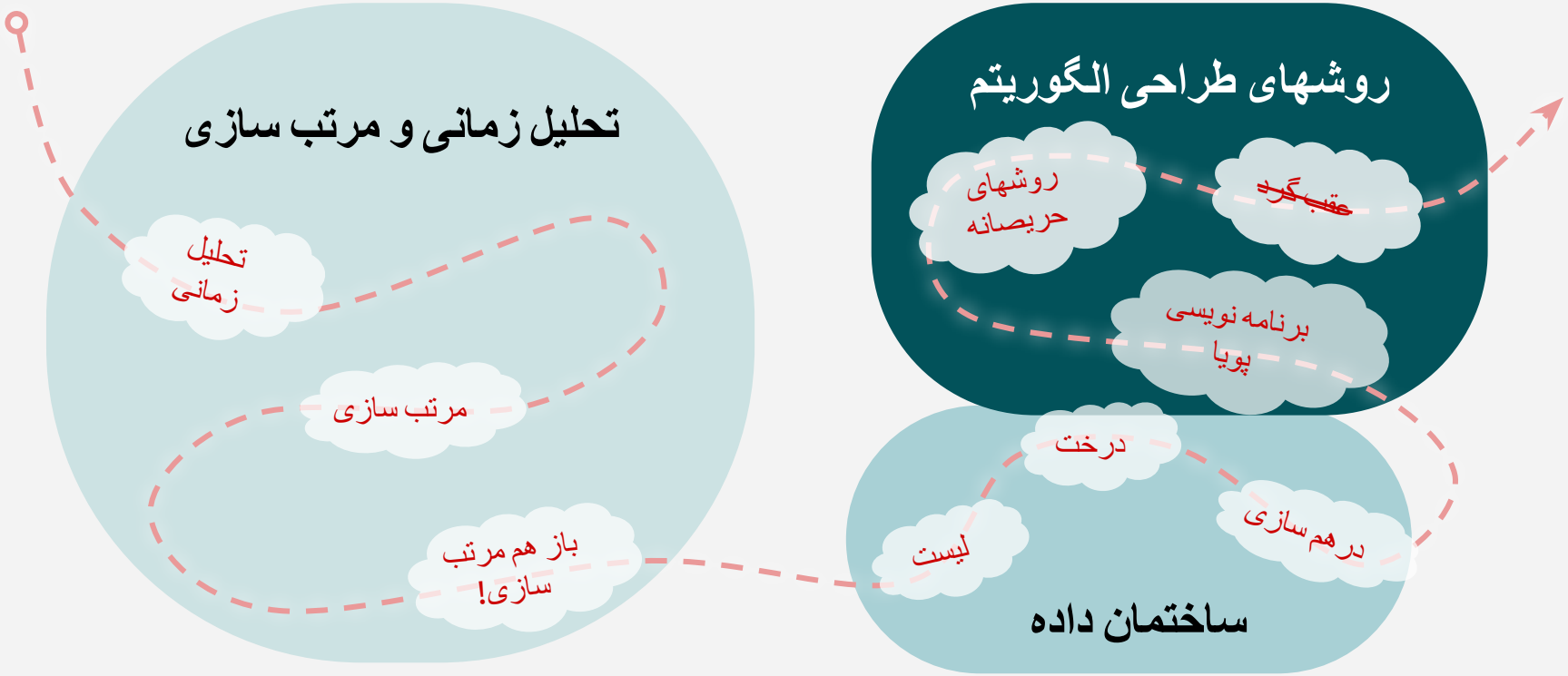
پاییز 1400

دوشنبه،13 دی 1400

اطلاع رسانی

- مهلت ارسال تمرین 4: ساعت 8 شب روز چهارشنبه (15 دی 1400)
- آخرین هفته (و جلسه!) کلاس
- هفته آینده: امتحان پایان ترم
- جلسه مراجعه مجازی برای رفع اشکال: روز جمعه، 17 دی 1400، ساعت 8 تا 9 شب
  - از طریق سامانه دروس (lmshome.aut.ac.ir) در قالب یک کلاس جبرانی

آنچه گذشت!


تحلیل زمانی و مرتب سازی
تحلیل زمانی
مرتب سازی
باز هم مرتب سازی!
روشهای طراحی الگوریتم
روشهای حریصانه
~~عقب گرد~~
برنامه نویسی پویا
درخت
لیست
درهم سازی
ساختمان داده

سوال 1 : قرینه کردن درخت

- تابعی بنویسید که درخت ورودی را قرینه می کند.

```
        1
      /   \
    2       3
  /   \
4       5
      /   \
    6       7
```

```
        1
      /   \
    3       2
          /   \
        5       4
      /   \
    7       6
```

سوال 1 : قرینه کردن درخت

```
Node mirror (Node root) {
    if (root == null)
        return null
    result = new Node(root.value)
    result.left = mirror(root.right)
    result.right = mirror(root.left)
}
```

سوال 1 : قرینه کردن درخت

```
Node mirror (Node root) {
    if (root == null)
        return null
    result = new Node(root.value)
    result.left = mirror(root.right)
    result.right = mirror(root.left)
}
```

$$T(n)=2T(n/2)+O(1)$$

سوال 1 : قرینه کردن درخت

```
Node mirror (Node root) {
    if (root == null)
        return null
    result = new Node(root.value)
    result.left = mirror(root.right)
    result.right = mirror(root.left)
}
```

T(n)=2T(n/2)+O(1)

Runtime: O(n)

سوال؟

سوال 2a : زمان اجرا

```
int happy (int n, int m) {
    if (n < 10) return n;
    else if (n < 100)
      return happy (n - 2, m);
    else
      return happy (n/2, m);
}
```

سوال 2a : زمان اجرا

```
int happy (int n, int m) {
    if (n < 10) return n;
    else if (n < 100)
      return happy (n - 2, m);
    else
      return happy (n/2, m);
}
```

$$T(n)=T(n/2)+O(1)$$

سوال 2a : زمان اجرا

```
int happy (int n, int m) {
    if (n < 10) return n;
    else if (n < 100)
      return happy (n - 2, m);
    else
      return happy (n/2, m);
}
```

T(n)=T(n/2)+O(1)

Runtime: O(log n)

سوال 2b : زمان اجرا

```
void sunny (int n) {
    j = 0;
    while (j < n) {
      for (int i = 0; i < n; ++i) {
          System.out.println("i = " + i);
          for (int k = 0; k < i; ++k)
             System.out.println("k = " + k);
      }
      j = j + 1;
    }
}
```

سوال 2b : زمان اجرا

```
void sunny (int n) {
    j = 0;
    while (j < n) {
      for (int i = 0; i < n; ++i) {
          System.out.println("i = " + i);
          for (int k = 0; k < i; ++k)
             System.out.println("k = " + k);
      }
      j = j + 1;
    }
}
```

Runtime: $O(n^3)$

سوال 2c : زمان اجرا

```
void funny (int n, int x) {
    for (int k = 0; k < 100; ++k)
        if (x > 500) {
            for (int i = 0; i < n * k; ++i)
                for (int j = 0; j < n; ++j)
                    System.out.println("x = " + x);
        }
    }
```

سوال 2c : زمان اجرا

```
void funny (int n, int x) {
    for (int k = 0; k < 100; ++k)
        if (x > 500) {
            for (int i = 0; i < n * k; ++i)
                for (int j = 0; j < n; ++j)
                    System.out.println("x = " + x);
        }
    }
```

Runtime: $O(n^2)$

سوال 2d : زمان اجرا

```
void smiley (int n) {
    for (int i = 0; i < n * n; ++i) {
        for (int k = 0; k < i; ++k)
            System.out.println("k = " + k);
        for (int j = n; j > 0; j--)
            System.out.println("j = " + j);
    }
}
```

سوال 2d : زمان اجرا

```
void smiley (int n) {
    for (int i = 0; i < n * n; ++i) {
        for (int k = 0; k < i; ++k)
            System.out.println("k = " + k);
        for (int j = n; j > 0; j--)
            System.out.println("j = " + j);
    }
}
```

Runtime: $O(n^4)$

سوال؟

سوال 3 : زیر بازه بزرگ

- ورودی: آرایه A[1..n] از اعداد مثبت و منفی
- آیا زیر بازه A[i..j] وجود دارد که مجموع آنها از 2 برابر بزرگترین عنصر A بزرگتر باشد؟

سوال 3 : زیر بازه بزرگ - تلاش اول

```
isLarge (A){
    goal = 2 * max(A)
    n = size(A)
    for (i = 1 ; i <= n ; i++)
        for (j = i ; j <= n ; j++) {
            sum = 0
            for (k = i ; k <= j ; k++)
                sum += A[k]
            if (sum > goal)
                return true
        }
    return false
}
```

سوال 3 : زیر بازه بزرگ - تلاش اول

```
isLarge (A){
    goal = 2 * max(A)
    n = size(A)
    for (i = 1 ; i <= n ; i++)
        for (j = i ; j <= n ; j++) {
            sum = 0
            for (k = i ; k <= j ; k++)
                sum += A[k]
            if (sum > goal)
                return true
        }
    return false
}
```

Runtime: $O(n^3)$

سوال 3 : زیر بازه بزرگ - تلاش دوم

```
isLarge (A){
    goal = 2 * max(A)
    n = size(A)
    cumSum = [0] * n
    for (i = 1 ; i <= n ; i++)
        cumSum[i] = cumSum[i-1] + A[i]
    for (i = 1 ; i <= n ; i++)
        for (j = i ; j <= n ; j++) {
            if ((cumSum[j]-cumSum[i-1]) > goal)
                return true
        }
    return false
}
```

سوال 3 : زیر بازه بزرگ - تلاش دوم

```
isLarge (A){
    goal = 2 * max(A)
    n = size(A)
    cumSum = [0] * n
    for (i = 1 ; i <= n ; i++)
        cumSum[i] = cumSum[i-1] + A[i]
    for (i = 1 ; i <= n ; i++)
        for (j = i ; j <= n ; j++) {
            if ((cumSum[j]-cumSum[i-1]) > goal)
                return true
        }
    return false
}
```

Runtime: $O(n^2)$

سوال 3 : زیر بازه بزرگ - تلاش سوم

```
isLarge (A){
    goal = 2 * max(A)
    n = size(A)
    cumSum = [0] * n
    for (i = 1 ; i <= n ; i++)
        cumSum[i] = cumSum[i-1] + A[i]
    return isLargeHelper(A, 1, n, goal, cumSum)
}
```

سوال 3 : زیر بازه بزرگ - تلاش سوم

```
isLargeHelper (A, i, j, goal, cumSum){
    if (i > j)        return false
    if (i == j)       return (A[i] > 2 x goal)
    mid = (i + j)/2
    if (  isLargeHelper(A, i, mid, goal, cumSum) or
          isLargeHelper(A, mid+1, j ,goal, cumSum))
              return true
    left = min(cumSum[i..mid])+1
    right = max(cumSum[mid+1..j])
    return ((cumSum[right]-cumSum[left-1]) > 2 x goal

}
```

سوال 3 : زیر بازه بزرگ - تلاش سوم - زمان اجرا

```
isLargeHelper (A, i, j, goal, cumSum){
    if (i > j)        return false
    if (i == j)       return (A[i] > 2 x goal)
    mid = (i + j)/2
    if (  isLargeHelper(A, i, mid, goal, cumSum) or
          isLargeHelper(A, mid+1, j ,goal, cumSum))
              return true
    left = min(cumSum[i..mid])+1
    right = max(cumSum[mid+1..j])
    return ((cumSum[right]-cumSum[left-1]) > 2 x goal
}
```

Runtime: T(n) = 2T(n/2) + O(n)

سوال 3 : زیر بازه بزرگ - تلاش سوم - زمان اجرا

```
isLargeHelper (A, i, j, goal, cumSum){
    if (i > j)        return false
    if (i == j)       return (A[i] > 2 x goal)
    mid = (i + j)/2
    if (  isLargeHelper(A, i, mid, goal, cumSum) or
          isLargeHelper(A, mid+1, j ,goal, cumSum))
              return true
    left = min(cumSum[i..mid])+1
    right = max(cumSum[mid+1..j])
    return ((cumSum[right]-cumSum[left-1]) > 2 x goal
}
```

Runtime: T(n) = 2T(n/2) + O(n)

Runtime: O(n log n)

سوال؟

سوال 4 : برگزاری امتحان

- مسئله:
  - هر دانشجو ابتدا باید هویت خود را برای ناظر امتحان احراز کند
  - هر دانشجو پس از احراز هویت، شروع به نوشتن امتحان میکند
  - تنها یک ناظر امتحان
  - افراد میتوانند همزمان با هم در حال نوشتن جواب باشند
- ورودی: n دانشجو که e_i زمان احراز هویت دانشجوی iام و w_i زمان نوشتن امتحان است
- خروجی: در بهترین ترتیب برای احراز هویت دانشجویان، کل امتحان چقدر طول میکشد

سوال 4 : برگزاری امتحان - ایده حریصانه

- نحوه انتخاب دانشجو: دانشجوی با مقدار بیشینه زمان نوشتن امتحان (w_i)

سوال 4 : برگزاری امتحان - ایده حریصانه - اثبات

- نحوه انتخاب دانشجو: دانشجوی با مقدار بیشینه زمان نوشتن امتحان (w_i)
- فرض کنید دو دانشجوی a و b به ترتیب زمان های (e_a،w_a) و (e_b،w_b) را دارند
- فرض کنید که $w_a > w_b$
- سبتدا دانشجوی b و سپس دانشجوی a:
- $total_{ba} = e_b + \max(w_b, e_a + w_a) = e_b + e_a + w_a$
- ابتدا دانشجوی a و سپس دانشجوی b:
- $total_{ab} = e_a + \max(w_a, e_b + w_b)$

سوال 4 : برگزاری امتحان - ایده حریصانه - اثبات

- نحوه انتخاب دانشجو: دانشجوی با مقدار بیشینه زمان نوشتن امتحان (w_i)
- فرض کنید دو دانشجوی a و b به ترتیب زمان های (e_a،w_a) و (e_b،w_b) را دارند
- فرض کنید که $w_a > w_b$
- سبتدا دانشجوی b و سپس دانشجوی a:
- $total_{ba} = e_b + \max(w_b, e_a + w_a) = e_b + e_a + w_a$
- ابتدا دانشجوی a و سپس دانشجوی b:
- $total_{ab} = e_a + \max(w_a, e_b + w_b)$
- حالت اول: $w_a > e_b + w_b$
- $total_{ab} = e_a + \max(w_a, e_b + w_b) = e_a + w_a < total_{ba}$
- حالت دوم: $w_a < e_b + w_b$
- $total_{ab} = e_a + \max(w_a, e_b + w_b) = e_a + e_b + w_b < e_a + e_b + w_a < total_{ba}$

سوال؟