ساختمان داده و الگوریتم ها (CE203)

جلسه بیست و یکم:
حل تمرین

سجاد شیرعلی شهرضا

پاییز 1400

دوشنبه، 22 آذر 1400

نمونه سوال از جدول درهم سازی

مسئله پیدا کردن پر تکرار ترین عنصر

- فرض های مسئله
  - آرایه X شامل n عدد مثبت
  - بزرگترین مقدار موجود در آرایه k است
- هدف: پیدا کردن عنصر با بیشتری تکرار
- دو روش مختلف:
  - استفاده از آرایه برای شمارش تکرار
  - استفاده از دیکشنری (بر مبنای جدول درهم سازی) برای شمارش تکرار
- سوال: پیدا کردن زمان اجرا در بدترین حالت و در حالت متوسط بر اساس n و k

روش اول: استفاده از آرایه

```
def frequentest_b(X):
    k = max(X)
    A = []
    for i in range(k + 1):
        A.append(0)
    best = X[0]
    for x in X:
        A[x] += 1
        if A[x] > A[best]:
            best = x
    return best
```

- استفاده از لیست در پایتون:
  - آرایه پویا (تغییر خودکار اندازه)

روش اول: استفاده از آرایه

```
def frequentest_b(X):
    k = max(X)
    A = []
    for i in range(k + 1):
        A.append(0)
    best = X[0]
    for x in X:
        A[x] += 1
        if A[x] > A[best]:
            best = x
    return best
```

- استفاده از لیست در پایتون:
  - آرایه پویا (تغییر خودکار اندازه)
- زمان اجرا در بدترین حالت؟

روش اول: استفاده از آرایه

```
def frequentest_b(X):
    k = max(X)
    A = []
    for i in range(k + 1):
        A.append(0)
    best = X[0]
    for x in X:
        A[x] += 1
        if A[x] > A[best]:
            best = x
    return best
```

- استفاده از لیست در پایتون:
  - آرایه پویا (تغییر خودکار اندازه)
- زمان اجرا در بدترین حالت؟
- $O(n^2)$

روش اول: استفاده از آرایه

```
def frequentest_b(X):
    k = max(X)
    A = []
    for i in range(k + 1):
        A.append(0)
    best = X[0]
    for x in X:
        A[x] += 1
        if A[x] > A[best]:
            best = x
    return best
```

- استفاده از لیست در پایتون:
  - آرایه پویا (تغییر خودکار اندازه)
- زمان اجرا در بدترین حالت؟
- $O(n^2)$
- زمان اجرا در حالت متوسط؟

روش اول: استفاده از آرایه

```
def frequentest_b(X):
    k = max(X)
    A = []
    for i in range(k + 1):
        A.append(0)
    best = X[0]
    for x in X:
        A[x] += 1
        if A[x] > A[best]:
            best = x
    return best
```

- استفاده از لیست در پایتون:
  - آرایه پویا (تغییر خودکار اندازه)
- زمان اجرا در بدترین حالت؟
- $O(n^2)$
- زمان اجرا در حالت متوسط؟
- $O(n)$

روش دوم: استفاده از دیکشنری

```
def frequentest_a(X):
    k = max(X)
    H = {}
    for x in X:
        H[x] = 0
    best = X[0]
    for x in X:
        H[x] += 1
        if H[x] > H[best]:
            best = x
    return best
```

- استفاده از دیکشنری در پایتون:
  - یک جدول درهم سازی
  - دارای یک مجموعه تابع درهم سازی سراسری
  - انتخاب تابع درهم سازی به صورت تصادفی

روش دوم: استفاده از دیکشنری

```
def frequentest_a(X):
    k = max(X)
    H = {}
    for x in X:
        H[x] = 0
    best = X[0]
    for x in X:
        H[x] += 1
        if H[x] > H[best]:
            best = x
    return best
```

- استفاده از دیکشنری در پایتون:
  - یک جدول درهم سازی
  - دارای یک مجموعه تابع درهم سازی سراسری
  - انتخاب تابع درهم سازی به صورت تصادفی
- زمان اجرا در حالت متوسط؟

روش دوم: استفاده از دیکشنری

```
def frequentest_a(X):
    k = max(X)
    H = {}
    for x in X:
        H[x] = 0
    best = X[0]
    for x in X:
        H[x] += 1
        if H[x] > H[best]:
            best = x
    return best
```

- استفاده از دیکشنری در پایتون:
  - یک جدول درهم سازی
  - دارای یک مجموعه تابع درهم سازی سراسری
  - انتخاب تابع درهم سازی به صورت تصادفی
- زمان اجرا در حالت متوسط؟
- $O(n+k)$

روش دوم: استفاده از دیکشنری

```
def frequentest_a(X):
    k = max(X)
    H = {}
    for x in X:
        H[x] = 0
    best = X[0]
    for x in X:
        H[x] += 1
        if H[x] > H[best]:
            best = x
    return best
```

- استفاده از دیکشنری در پایتون:
  - یک جدول درهم سازی
  - دارای یک مجموعه تابع درهم سازی سراسری
  - انتخاب تابع درهم سازی به صورت تصادفی
- زمان اجرا در حالت متوسط؟
- $O(n+k)$
- زمان اجرا در بدترین حالت؟

روش دوم: استفاده از دیکشنری

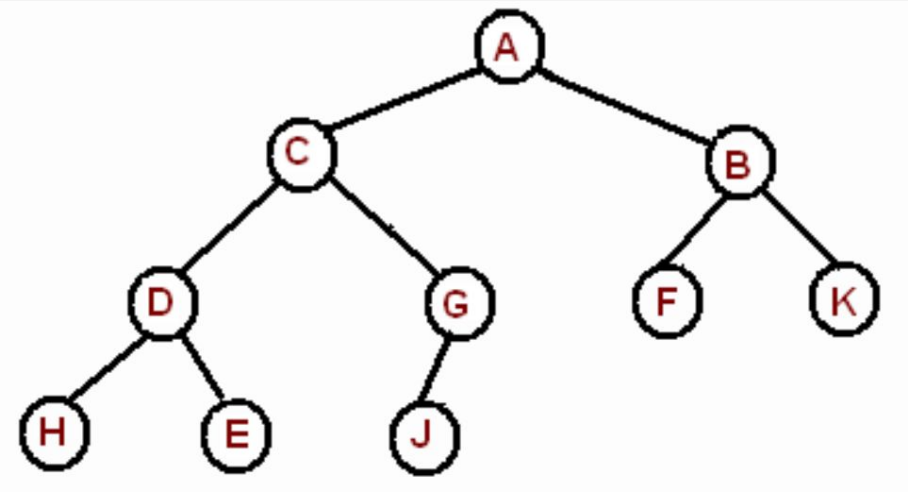
```
def frequentest_a(X):
    k = max(X)
    H = {}
    for x in X:
        H[x] = 0
    best = X[0]
    for x in X:
        H[x] += 1
        if H[x] > H[best]:
            best = x
    return best
```

- استفاده از دیکشنری در پایتون:
  - یک جدول درهم سازی
  - دارای یک مجموعه تابع درهم سازی سراسری
  - انتخاب تابع درهم سازی به صورت تصادفی
- زمان اجرا در حالت متوسط؟
- $O(n+k)$
- زمان اجرا در بدترین حالت؟
- $O(n+k)$

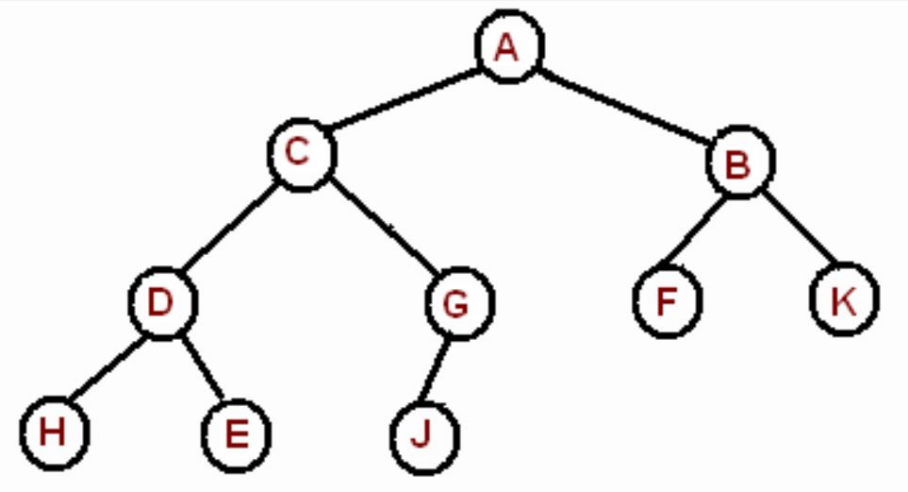
نمونه سوال از صف اولویت

- آرایه زیر نمایش یک صف اولویت کمینه (min-heap) است:
- [A, C, B, D, G, F, K, H, E, J]
- محل کوچکترین عنصر؟

- آرایه زیر نمایش یک صف اولویت کمینه (min-heap) است:
- [A, C, B, D, G, F, K, H, E, J]
- محل کوچکترین عنصر؟

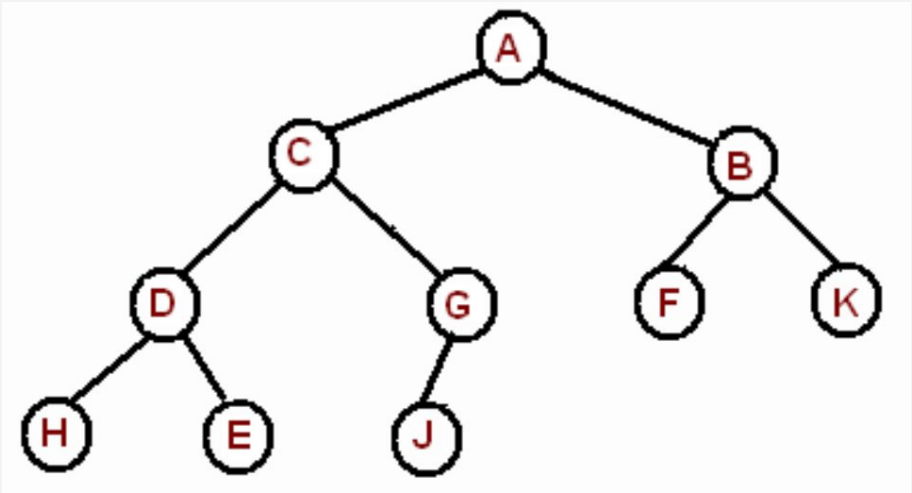


- آرایه زیر نمایش یک صف اولویت کمینه (min-heap) است:
- [A, C, B, D, G, F, K, H, E, J]



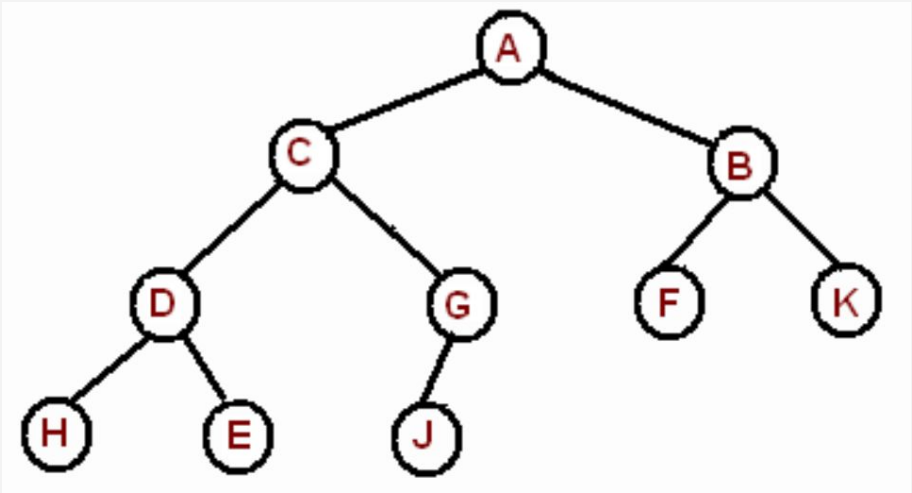
- محل کوچکترین عنصر؟
  - A ○

- آرایه زیر نمایش یک صف اولویت کمینه (min-heap) است:
- [A, C, B, D, G, F, K, H, E, J]



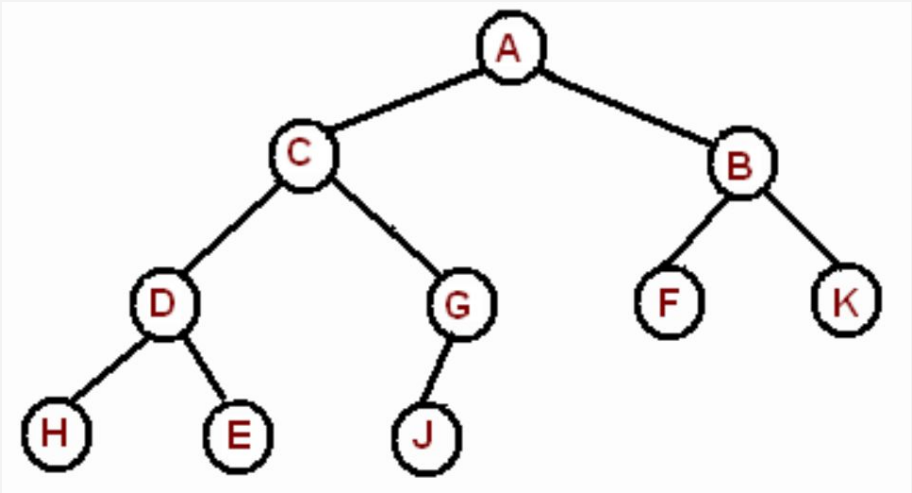
- محل کوچکترین عنصر؟
  - A
- محل سومین عنصر کوچک؟

- آرایه زیر نمایش یک صف اولویت کمینه (min-heap) است:
- [A, C, B, D, G, F, K, H, E, J]



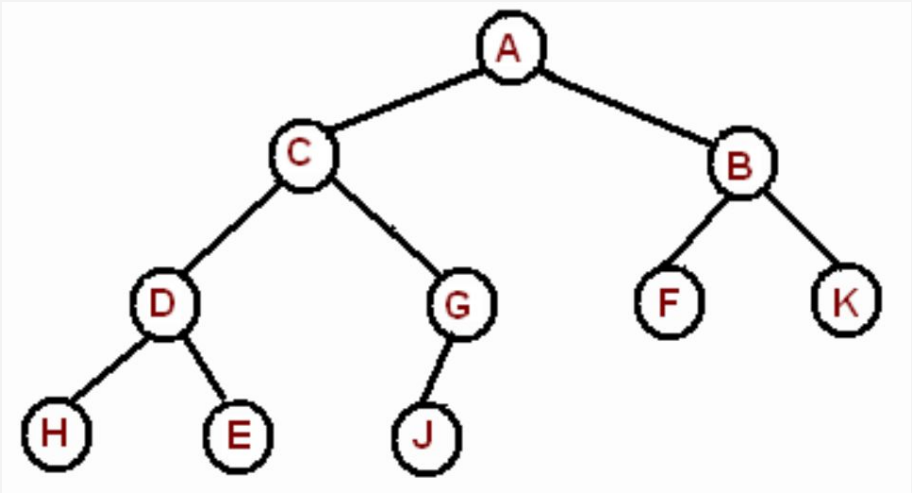
- محل کوچکترین عنصر؟
  - A
- محل سومین عنصر کوچک؟
  - یکی از B, C, D, G, F, K
  - اشتباه رایج: حتما در B یا C است

- آرایه زیر نمایش یک صف اولویت کمینه (min-heap) است:
- [A, C, B, D, G, F, K, H, E, J]



- محل کوچکترین عنصر؟
  - A
- محل سومین عنصر کوچک؟
  - یکی از B, C, D, G, F, K
  - اشتباه رایج: حتما در B یا C است
- محل بزرگترین عضو؟

- آرایه زیر نمایش یک صف اولویت کمینه (min-heap) است:
- [A, C, B, D, G, F, K, H, E, J]



- محل کوچکترین عنصر؟
  - A
- محل سومین عنصر کوچک؟
  - یکی از B, C, D, G, F, K
  - اشتباه رایج: حتما در B یا C است
- محل بزرگترین عضو؟
  - یکی از F, K, H, E, J
  - اشتباه رایج: بزرگتین عنصر در آخرین سطح است یعنی یکی از H, E, J

انتخاب مرتب سازی مناسب

جایزه بهترین دانشجو

- قرار است برنامه ای برای رتبه بندی دانشجویان دانشکده در زمینه های مختلف بنویسیم
- در دانشکده n دانشجو داریم
- معیار انتخاب در رشته های مختلف، متفاوت است
- هدف: در هر رشته، سریع ترین روش برای مرتب سازی دانشجویان بر اساس آن معیار

رشته روابط عمومی

- معیار: تعداد دوست در بین دانشجویان
  - محاسبه تعداد دوستان هر دانشجو به زمان ثابت ($O(1)$) نیاز دارد

رشته روابط عمومی (جواب)

- معیار: تعداد دوست در بین دانشجویان
  - محاسبه تعداد دوستان هر دانشجو به زمان ثابت (O(1)) نیاز دارد

- تعداد دوستان هر نفر عددی بین 0 تا n-1 است
- از روش مرتب سازی شمارشی (و یا مبنایی) استفاده میکنیم
- زمان لازم: O(n)

رشته قوی ترین دانشجو

- تعریف قوی تر بودن: توانایی شکست حریفان در مسابقه مچ انداختن
- نحوه بررسی: مسابقه رودرروی مچ اندازی بین دو دانشجو که زمان ثابت ($O(1)$) طول میکشد

رشته قوی ترین دانشجو (جواب)

- تعریف قوی تر بودن: توانایی شکست حریفان در مسابقه مچ انداختن
- نحوه بررسی: مسابقه رودرروی مچ اندازی بین دو دانشجو که زمان ثابت ($O(1)$) طول میکشد

- هر مسابقه مچ انداختن، یک مقایسه بین دو دانشجو است
- کران پایین مرتب سازی مبتنی بر مقایسه برقرار است $\Omega(n \lg n)$
- هر الگوریتم $O(n \lg n)$ خوب است (مثلا مرتب سازی ادغامی)

سوال؟