Selecting Problem

طراحی الگوریتم‌ها – جلسه یازدهم

Introduction to Algorithm

استاد: جوانمردی

۱۳۹۹/۸/۱۰

مرور جلسه قبل

مرتب‌سازی خطی Radix Sort

اطلاعات مازاد: اعداد دارای d رقم و هر رقم k مقدار متفاوت

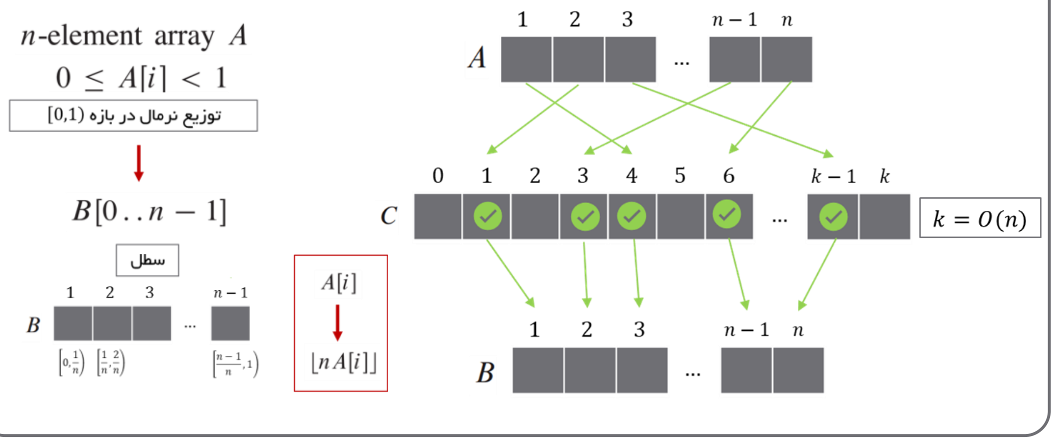
329	720	720	329
457	355	329	355
657	436	436	436
839	457	839	457
436	657	355	657
720	329	457	720
355	839	657	839

RADIX-SORT(A, d)

1 **for** $i = 1$ **to** d
2 use a stable sort to sort array A on digit i

- زمان اجرا: $\theta(d(n+k))$
- اگر $k = O(n)$ و $d = O(1)$ باشد زمان اجرای radix sort $\theta(n)$ خواهد بود

مرتب‌سازی سطلی Bucket Sort



تحلیل زمانی Radix Sort

- اگر n عدد b بیتی داشته باشیم، برای هر عدد مثبت دلخواه $r \le b$ خواهیم داشت:

زمان اجرایی radix sort برای این اعداد $\theta((b/r)(n+2^r))$ خواهد بود

به شرطی که مرتب‌سازی پایدار استفاده شده $\theta(n+k)$ باشد

برای اعداد n و b تعیین $r \le b$ بگونه‌ای که زمان اجرای $(b/r)(n+2^r)$ را کمینه کند

حالت اول: $b < \lfloor \lg n \rfloor$ ← برای $r = b$ خواهیم داشت: $\Theta(n)$

حالت دوم: $b \ge \lfloor \lg n \rfloor$ ← برای $r = \lfloor \lg n \rfloor$ خواهیم داشت: $\Theta(bn/\lg n)$

درصورتی که $b = O(\lg n)$ باشد با انتخاب $r = \lfloor \lg n \rfloor$ زمان اجری radix sort برابر $O(n)$

تحلیل زمانی Bucket Sort

BUCKET-SORT(A)

1 let $B[0 \,..\, n-1]$ be a new array
2 $n = A.length$
3 **for** $i = 0$ **to** $n-1$
4 make $B[i]$ an empty list
5 **for** $i = 1$ **to** n
6 insert $A[i]$ into list $B[\lfloor nA[i] \rfloor]$
7 **for** $i = 0$ **to** $n-1$
8 sort list $B[i]$ with insertion sort
9 concatenate the lists $B[0], B[1], \ldots, B[n-1]$ together in order

$$T(n) = \Theta(n) + \sum_{i=0}^{n-1} O(n_i^2)$$

n_i متغیر تصادفی نشانگر تعداد المان‌ها در سطل B_i

$$= \Theta(n) + \sum_{i=0}^{n-1} O\left(2 - \frac{1}{n}\right) = \Theta(n)$$

فصل ۹ کتاب

- میانه و مرتبه‌های آماری
  - کمینه و بیشینه
  - امید زمان اجرای الگوریتم انتخاب
  - بدترین زمان اجرای الگوریتم انتخاب

مرتبه‌های آماری

- انتخاب i امین کوچکترین عنصر در یک آرایه عددی n عنصری
- کمینه اعداد: اولین مرتبه آماری
- بیشینه اعداد: n امین مرتبه آماری
- میانه اعداد: عدد وسطی آرایه
  - اگر n فرد باشد: عنصر $(n+1)/2$ ام
  - اگر n زوج باشد: عنصرهای $n/2$ و $\frac{n}{2}+1$
  - حالت کلی: میانه پایین $\lfloor (n+1)/2 \rfloor$ و میانه بالا $\lceil (n+1)/2 \rceil$
- میانه در این درس برای سادگی میانه پایین فرض می‌شود
- اعداد آرایه یکتا هستند: هیچ دو عدد تکراری نداریم (صرفا برای راحتی، قابل تعمیم به حالت کلی)

Selection Problem مسئله انتخاب

Input: A set A of n (distinct) numbers and an integer i, with $1 \leq i \leq n$.

Output: The element $x \in A$ that is larger than exactly $i - 1$ other elements of A.

- ورودی: مجموعه A با n عدد یکتا و یک عدد طبیعی i به شرط $1 \leq i \leq n$
- خروجی: عنصر $x \in A$ به شرطی که x دقیقا از $i-1$ عنصر دیگر مجموعه A بزرگتر باشد

- راحل حل $O(n\lg n)$: مرتب‌سازی آرایه با $O(n\lg n)$ و انتخاب i امین عنصر آرایه با $O(1)$
- هدف: ارائه الگوریتم سریع‌تر

انتخاب کمینه یا بیشینه

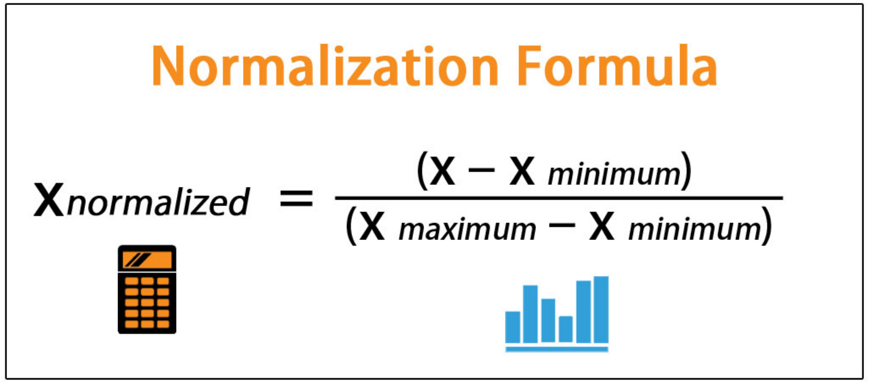
- چه تعداد مقایسه برای تعیین کمینه یا بیشینه مورد نیاز است؟
- حد بالا: شروع از اولین عنصر و انجام $n-1$ مقایسه و نگهداری کمینه یا بیشینه

```
MINIMUM(A)
min = A[1]
for i = 2 to A.length
    if min > A[i]
        min = A[i]
return min
```

- آیا این بهترین کار است؟
- جواب: بله! می‌توان نشان داد که $n-1$ مقایسه حد پایین هم هست
- تورنومنت اعداد: هرکسی حداقل یک باخت!

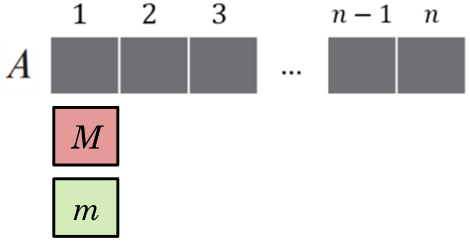
انتخاب کمینه و بیشینه بصورت همزمان

- در برخی موارد نیاز به پیدا کردن همزمان ماکزیمم و مینیمم داریم
- مثال: نرمال کردن مجموعه ای از اعداد



asymptotically optimal

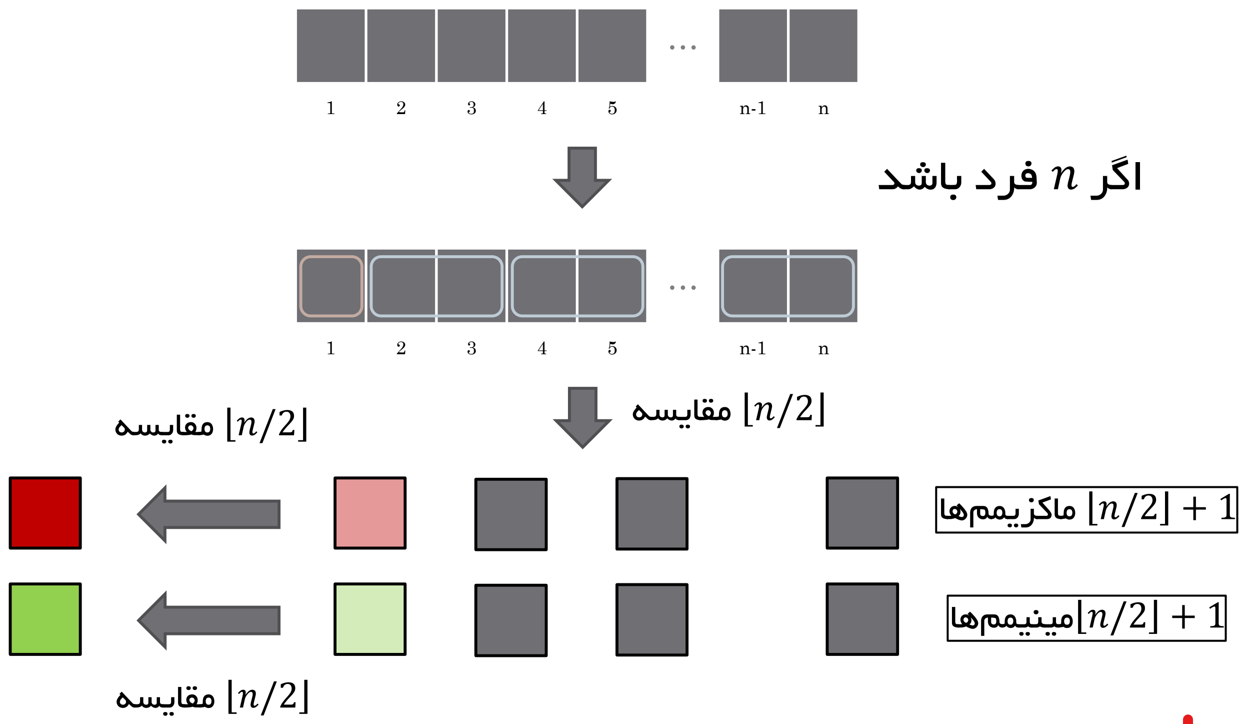
- جواب $\Theta(n)$: تعیین مستقل مینیمم و ماکزیمم هرکدام $n-1$ مقایسه، مجموعا $2n-2$ مقایسه



- روش بهتر؟ جواب: بله! ⟵ تعیین همزمان مینیمم و ماکزیمم با حداکثر $3\lfloor n/2 \rfloor$ مقایسه
- به جای مقایسه هر عنصر با مینیمم و ماکزیمم فعلی ⟵ مقایسه عناصر بصورت جفت

انتخاب کمینه و بیشینه بصورت همزمان

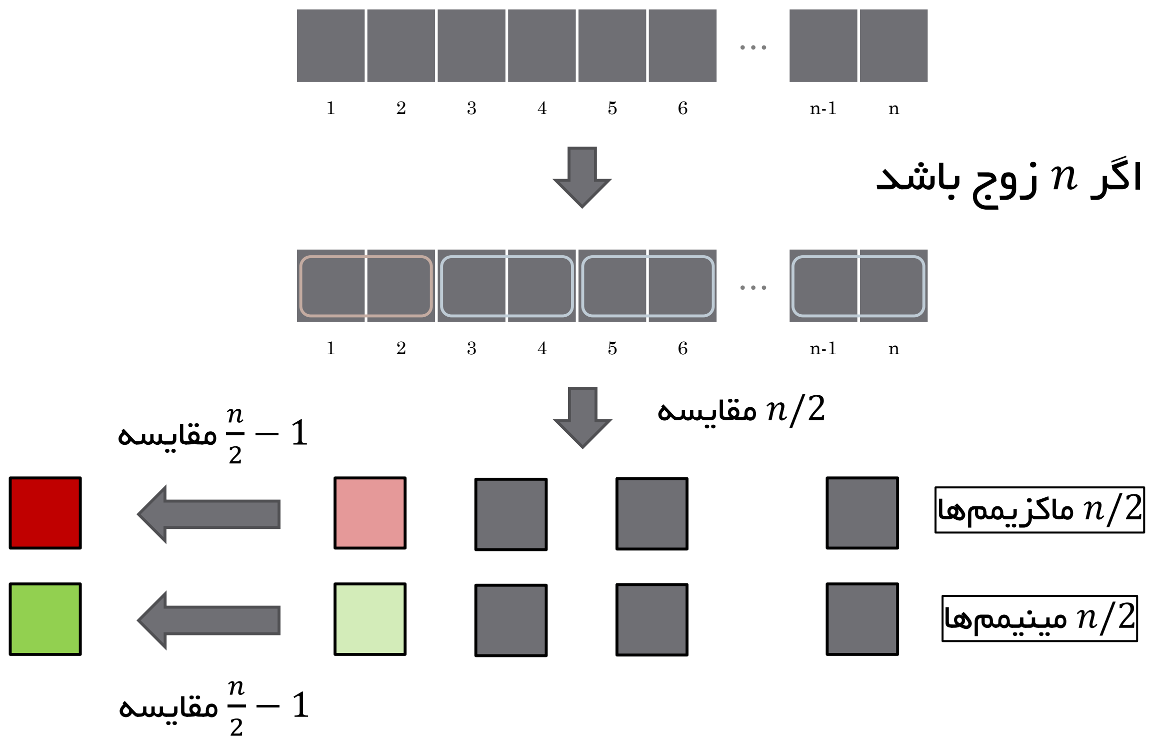
- به جای مقایسه هر عنصر با مینیمم و ماکزیمم فعلی ⟵ مقایسه عناصر بصورت جفت



$\boxed{\text{مجموعا } 3\lfloor n/2 \rfloor \text{ مقایسه}}$

انتخاب کمینه و بیشینه بصورت همزمان

- به جای مقایسه هر عنصر با مینیمم و ماکزیمم فعلی ⟵ مقایسه عناصر بصورت جفت

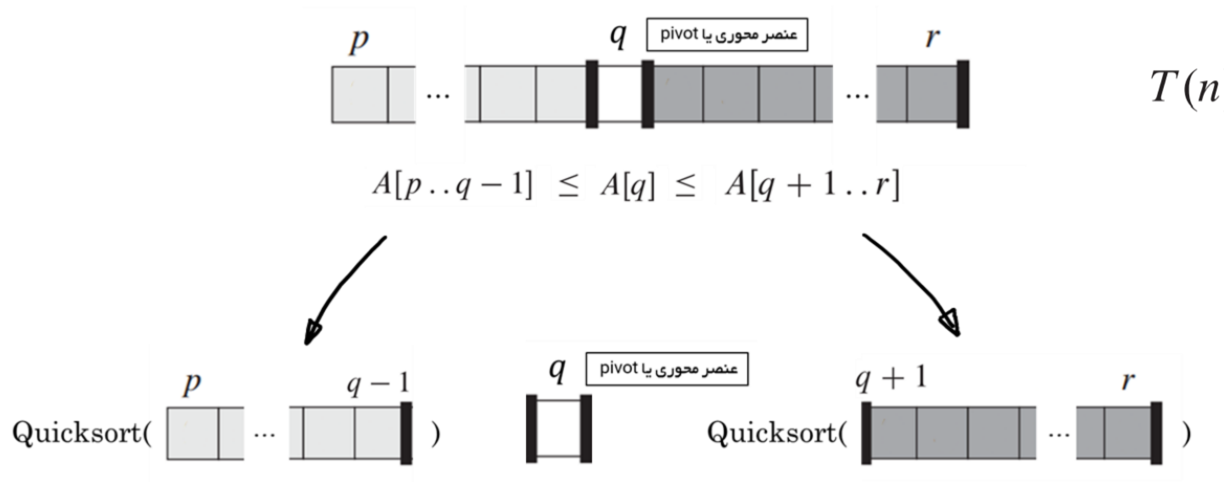


مجموعا $\frac{3n}{2}-2$ مقایسه ⟵ حداکثر $3\lfloor n/2 \rfloor$ مقایسه

مسئله انتخاب با زمان اجرای متوسط خطی

- ورودی: مجموعه A با n عدد یکتا و یک عدد طبیعی i به شرط $1 \le i \le n$
- خروجی: عنصر $x \in A$ به شرطی که x دقیقا از $i-1$ عنصر دیگر مجموعه A بزرگتر باشد

- راه حل با زمان متوسط $\Theta(n)$: روش تقسیم و حل مشابه ایده اصلی مرتب‌سازی سریع



$$T(n) = (T(q) + T(n-q-1)) + \Theta(n)$$

$$Expected\ Running\ Time: \Theta(n\lg n)$$

$$Expected\ Running\ Time: \Theta(n)$$

دارای q عنصر

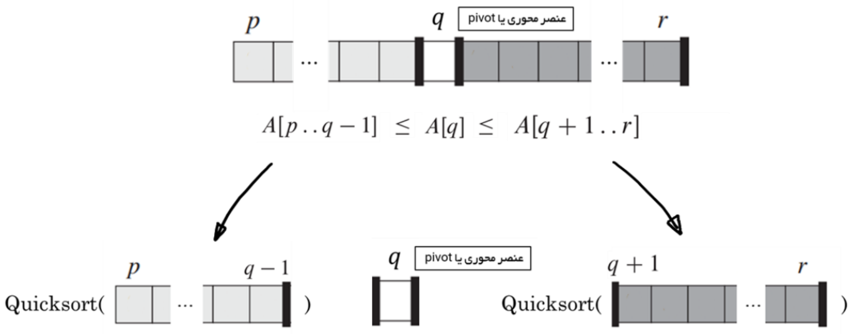
دارای $n-q-1$ عنصر

مسئله انتخاب با زمان اجرای متوسط خطی

- ورودی: مجموعه A با n عدد یکتا و یک عدد طبیعی i به شرط $1 \leq i \leq n$
- خروجی: عنصر $x \in A$ به شرطی که x دقیقا از $i-1$ عنصر دیگر مجموعه A بزرگتر باشد

```
RANDOMIZED-SELECT(A, p, r, i)
if p == r
    return A[p]
q = RANDOMIZED-PARTITION(A, p, r)
k = q - p + 1
if i == k            // the pivot value is the answer
    return A[q]
elseif i < k
    return RANDOMIZED-SELECT(A, p, q - 1, i)
else return RANDOMIZED-SELECT(A, q + 1, r, i - k)
```

```
RANDOMIZED-PARTITION(A, p, r)
i = RANDOM(p, r)
exchange A[r] with A[i]
return PARTITION(A, p, r)
```



دارای q عنصر

دارای $n-q-1$ عنصر

1. $K = q - p - 1 = i$
pivot برابر i امین عنصر است

2. $K = q - p - 1 > i$
در نیمه اول دنبال i امین میگردیم

3. $K = q - p - 1 \leq i$
در نیمه دوم دنبال i-k امین میگردیم

تحلیل زمانی RANDOMIZED-SELECT

- بدترین زمان اجرا حتی برای پیدا کردن مینیمم: $\Theta(n^2)$
- زمان اجرای متوسط: $\Theta(n)$
- زمان اجرای تعیین i امین عنصر از آرایه $A[p..r]$ یک متغیر تصادفی: $T(n)$
- هدف: محاسبه $\mathrm{E}[T(n)]$
- استفاده از Randomized-partition:
  - احتمال اینکه آرایه $A[p..q]$ برای همه مقادیر k در محدوده $1 \le k \le n$ دارای k عنصر باشد برابر با $1/n$

$$X_k = \mathrm{I}\{\text{the subarray } A[p \,..\, q] \text{ has exactly } k \text{ elements}\}$$

$$\mathrm{E}[X_k] = 1/n$$

آمار و احتمالات

متغیر تصادفی شاخص

$$I(A) = \begin{cases} 1, & \text{if } A \text{ happen} \\ 0, & \text{if } A \text{ not happen} \end{cases}$$

تحلیل زمانی RANDOMIZED-SELECT

3	2	1
$k \le i$	$k > i$	$k = i$
در نیمه دوم دنبال i-k امین میگردیم	در نیمه اول دنبال i امین میگردیم	pivot برابر i امین عنصر است
تکرار برای $n - k$ عنصر	تکرار برای $k - 1$ عنصر	تمام!
T(n-K)	T(K-1)	T(1)

$$T(n) \le \sum_{k=1}^{n} X_k \cdot (T(\max(k-1, n-k)) + O(n))$$

$$= \sum_{k=1}^{n} X_k \cdot T(\max(k-1, n-k)) + O(n) .$$

تحلیل زمانی RANDOMIZED-SELECT

- محاسبه امید ریاضی زمان اجرا یا زمان اجرای متوسط:

$$
\begin{aligned}
\mathrm{E}[T(n)] & \\
&\le \mathrm{E}\left[\sum_{k=1}^{n} X_k \cdot T(\max(k-1, n-k)) + O(n)\right] \\
&= \sum_{k=1}^{n} \mathrm{E}[X_k \cdot T(\max(k-1, n-k))] + O(n) \\
&= \sum_{k=1}^{n} \mathrm{E}[X_k] \cdot \mathrm{E}[T(\max(k-1, n-k))] + O(n) \\
&= \sum_{k=1}^{n} \frac{1}{n} \cdot \mathrm{E}[T(\max(k-1, n-k))] + O(n)
\end{aligned}
$$

ویژگی خطی بودن

ضرب متغیر مستقل

امیدریاضی و برخی خواص آن

تعریف امیدریاضی برای متغیر تصادفی x_i: $E(X) = \sum_{i=1}^{n} p_i x_i$

ویژگی خطی بودن امیدریاضی: $E(aX + b) = aE(X) + b$

برای متغیر تصادفی مستقل X و Y داریم: $E(XY) = E(X)E(Y)$

تحلیل زمانی RANDOMIZED-SELECT

• ادامه محاسبات:

$$\mathrm{E}\left[T(n)\right] = \sum_{k=1}^{n} \frac{1}{n} \cdot \mathrm{E}\left[T(\max(k-1, n-k))\right] + O(n)$$

$$\max(k-1, n-k) = \begin{cases} k-1 & \text{if } k > \lceil n/2 \rceil \,, \\ n-k & \text{if } k \le \lceil n/2 \rceil \,. \end{cases}$$

1	2	3	4	5	6	7	8	9	10
k-1	k-1	k-1	k-1	k-1	n-k	n-k	n-k	n-k	n-k
0	1	2	3	4	4	3	2	1	0

$$\mathrm{E}\left[T(n)\right] \le \frac{2}{n} \sum_{k=\lfloor n/2 \rfloor}^{n-1} \mathrm{E}\left[T(k)\right] + O(n)$$

برای زوج

$k=$	1	2	3	4	5	6	7	8	9	10	11
$k-1=$	0	1	2	3	4	5	6	7	8	9	10
$n-k$	10	9	8	7	6	5	4	3	2	1	0

$n=11$

تحلیل زمانی RANDOMIZED-SELECT

$$\mathrm{E}[T(n)] \le \frac{2}{n}\sum_{k=\lfloor n/2\rfloor}^{n-1} \mathrm{E}[T(k)] + O(n) \longrightarrow \mathrm{E}[T(n)] = O(n)$$

جایگذاری

$$\mathrm{E}[T(n)] \le cn$$

حکم

$$\begin{aligned}\mathrm{E}[T(n)] &\le \frac{2}{n}\sum_{k=\lfloor n/2\rfloor}^{n-1} ck + an \\ &= \frac{2c}{n}\left(\sum_{k=1}^{n-1} k - \sum_{k=1}^{\lfloor n/2\rfloor-1} k\right) + an\end{aligned}$$

برای nهای کوچکتر از مقداری

$$T(n) = O(1)$$

تحلیل زمانی RANDOMIZED-SELECT

$$E[T(n)] \le cn$$ حکم

$$\begin{aligned}
E[T(n)] &= \frac{2c}{n}\left(\sum_{k=1}^{n-1} k - \sum_{k=1}^{\lfloor n/2 \rfloor - 1} k\right) + an \\
&= \frac{2c}{n}\left(\frac{(n-1)n}{2} - \frac{(\lfloor n/2 \rfloor - 1)\lfloor n/2 \rfloor}{2}\right) + an \\
&\le \frac{2c}{n}\left(\frac{(n-1)n}{2} - \frac{(n/2-2)(n/2-1)}{2}\right) + an \\
&= c\left(\frac{3n}{4} + \frac{1}{2} - \frac{2}{n}\right) + an \\
&\le \frac{3cn}{4} + \frac{c}{2} + an \\
&= cn - \left(\frac{cn}{4} - \frac{c}{2} - an\right)
\end{aligned}$$

$$n \ge \frac{c/2}{c/4 - a} = \frac{2c}{c - 4a}$$

$$T(n) = O(1) \text{ for } n < 2c/(c-4a)$$

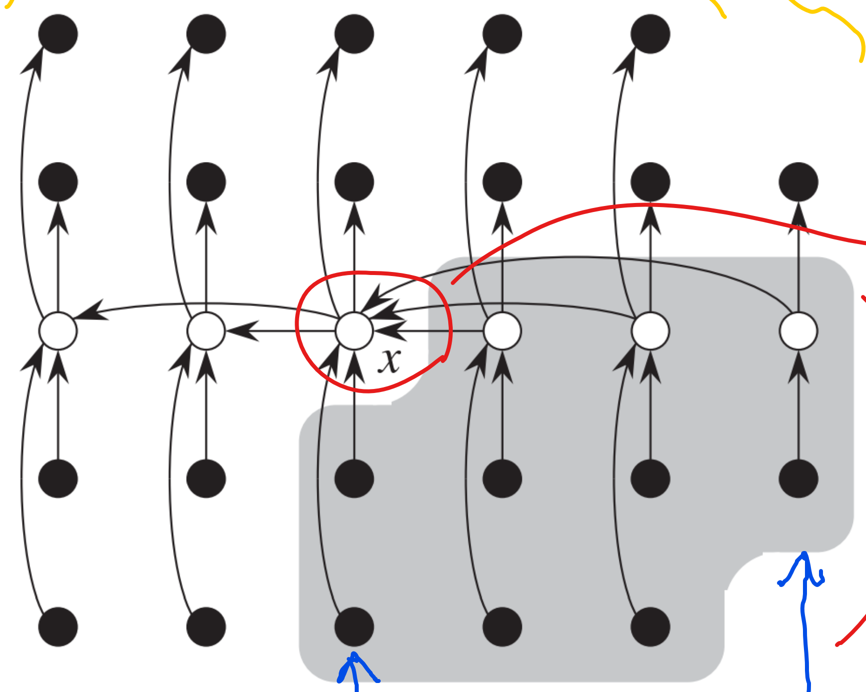
$$E[T(n)] = O(n)$$

مسئله انتخاب با بدترین زمان اجرای خطی

- ورودی: مجموعه A با n عدد یکتا و یک عدد طبیعی i به شرط $1 \leq i \leq n$
- خروجی: عنصر $A \ \ni x$ به شرطی که x دقیقا از $i-1$ عنصر دیگر مجموعه A بزرگتر باشد
- ایده اصلی: مشابه روش قبل (recursive portioning) اما با ضمانت تقسیم بندی خوب!

- مراحل تابع SELECT
- مرحله ۱: تقسیم ورودی به $\lceil n/5 \rceil$ گروه متشکل از ۵ عنصر و حداکثر یک گروه مابقی اعداد
- مرحله ۲: یافتن میانه در هر $\lceil n/5 \rceil$ گروه – مرتب‌سازی درجی و انتخاب عنصر میانه
- مرحله ۳: با تابع SELECT بصورت بازگشتی میانه‌ی میانه‌ها را از $\lceil n/5 \rceil$ میانه انتخاب میکنیم
- مرحله ۴: Partition را با استفاده از میانه‌ی میانه‌ها x انجام بده (x برابر k امین عنصر)
- مرحله ۵: اگر $i=k$ پس x جواب است. اگر $i<k$ در سمت پایین و $i>k$ در سمت بالا دنبال عدد بگرد

مسئله انتخاب با بدترین زمان اجرای خطی



- دایره: اعداد داخل آرایه
- ستون: گروه‌های ۵ عنصری
- دایره سفید: میانه هر گروه
- x: میانه‌ی میانه‌ها
- پیکان: از بزرگ‌تر به کوچک‌تر
- ناحیه خاکستری: اعداد بزرگتر از x

- حداقل نیمی از میانه‌ها بزرگتر یا مساوی میانه‌ی میانه‌ها x هستند
- حداقل نیمی از $\lceil n/5 \rceil$ گروه حداقل ۳ عنصر بزرگتر از x دارند
- گروه کوچک و گروه خود x را استثنا میکنیم (حداقل)
- بالعکس آن هم صادق است! حداقل همین تعداد کوچکتر داریم

$$3\left(\left\lceil \frac{1}{2}\left\lceil \frac{n}{5} \right\rceil \right\rceil - 2\right) \geq \boxed{\frac{3n}{10} - 6}$$

مسئله انتخاب با بدترین زمان اجرای خطی

- مرحله ۱ : تقسیم ورودی به $\lceil n/5 \rceil$ گروه متشکل از ۵ عنصر و حداکثر یک گروه مابقی اعداد $\quad O(n)$
- مرحله ۲: یافتن میانه در هر $\lceil n/5 \rceil$ گروه – مرتب‌سازی درجی و انتخاب عنصر میانه $\quad O(n)$
- مرحله ۳: با تابع SELECT بصورت بازگشتی میانه‌ی میانه‌ها را از $\lceil n/5 \rceil$ میانه انتخاب میکنیم $\quad T(\lceil n/5 \rceil)$
- مرحله ۴: Partition را با استفاده از میانه‌ی میانه‌ها x انجام بده (x برابر k امین عنصر) $\quad O(n)$
- مرحله ۵: اگر $i = k$ پس x جواب است. اگر $i = k$ در سمت پایین و $i = k$ در سمت بالا دنبال عدد بگرد $\quad T(7n/10 + 6)$

$(n - (\frac{3n}{10} - 6))$

$$T(n) = T(\lceil n/5 \rceil) + T(7n/10 + 6) + O(n)$$

مسئله انتخاب با بدترین زمان اجرای خطی

$$T(n) = T(\lceil n/5 \rceil) + T(7n/10 + 6) + O(n)$$

$$\begin{aligned} T(n) &\le c\lceil n/5 \rceil + c(7n/10 + 6) + an \\ &\le cn/5 + c + 7cn/10 + 6c + an \\ &= 9cn/10 + 7c + an \\ &= cn + (-cn/10 + 7c + an)\,, \end{aligned}$$

$$n \ge 140$$

$$\boxed{-cn/10 + 7c + an \le 0} \qquad c \ge 10a(n/(n-70)) \qquad n/(n-70) \le 2$$

$$c \ge 20a$$

$$T(n) \le \begin{cases} O(1) & \text{if } n < 140 \\ T(\lceil n/5 \rceil) + T(7n/10 + 6) + O(n) & \text{if } n \ge 140 \end{cases}$$