روش‌های طراحی الگوریتم‌ها

دسته‌بندی

۱) **مبتنی بر استقرا** (induction)

۲) **تقسیم‌وحل** (divide and conquer)

۳) **برنامه‌ریزی پویا** (dynamic programming)

۴) **حریصانه** (greedy)

۵) **جست‌وجوی فضای حالت** (state space search)

- ◁ **پس‌گرد** (backtracking)
- ◁ **درخت بازی** game tree، **هرس** $\alpha - \beta$
- ◁ **انشعاب و حد** (branch and bound)

۶) **روش‌های تقریبی** (approximation) **یا مکاشفه‌ای** (heuristic)

۷) **الگوریتم‌های تصادفی** (Randomized)

طراحی الگوریتم با استقرا

مسئله: «ستاره‌ی مشهور» (celebrity)

می‌خواهیم با حداقل تعداد پرسش از n نفر، فردی که ویژگی‌های یک «ستاره‌ی مشهور» را دارد، در صورت وجود، پید کنیم.

یک نفر ستاره است اگر بقیه او را از قبل بشناسند و او هیچ‌کس را نشناسد.

از گراف «شناختن» بی‌اطلاع هستیم.

اما مجازیم از a بپرسیم که آیا b را می‌شناسد؟ این یک پرسش است.

چون $n(n-۱)/۲$ گروه دو نفری داریم پس اگر پرسش‌ها به طور دل‌خواه باشند، در بدترین حالت نیاز به $n(n-۱)$ پرسش داریم.

راه حل اول:

فرض: می‌توانیم ستاره را بین $n-1$ نفر اول توسط استقرا به‌دست آوریم.

حداکثر یک ستاره داریم $\Longleftarrow$ سه حالت ممکن است:

۱) ستاره بین $n-1$ نفر اول است.

۲) نفر nام ستاره است.

۳) ستاره نداریم.

در حالت اول فقط باید بررسی کنیم که نفر nام ستاره را می‌شناسد و ستاره او را نمی‌شناسد.

در دو حالت بعد، حداکثر به $۲(n-۱)$ پرسش نیاز داریم؛ چرا که باید روشن کنیم که آیا هریک از $n-۱$ نفر بقیه نفر nام می‌شناسد و نفر nام او را نمی‌شناسد.

بنابراین، جمع کل پرسش‌ها $n(n-۱)$ است که در بدترین حالت هم به آن رسیدیم.

راه‌حل بهتر:

به روش حذفی عمل می‌کنیم، در هر پرسش یک نفر را از مجموعه حذف کنیم و با استفاده از استقرا راه حل را دنبال می‌کنیم.

از A بپرسیم که آیا B را می‌شناسد یا نه؟

جواب مثبت $\Longleftarrow$ A نمی‌تواند ستاره باشد

اگر جواب منفی $\Longleftarrow$ B نمی‌تواند ستاره باشد

با $n-۱$ پرسش به یک نفر به نام s می‌رسیم؛ تنها s ممکن است ستاره باشد.

با $۲(n-۱)$ پ‌رسش این امر را می‌توان مشخص کرد.

$$n-۱+۲(n-۱)=۳(n-۱)$$

این تعداد پرسش‌ها کمینه نیست!

می‌توان طوری سوال کرد که مطمئناً از s قبلاً $\frac{\lg n}{۲}$ سؤال پرسیده باشیم. پس این تعداد کم می‌شود.

پیاده‌سازی

ورودی `Know` **یک ماتریس مجاورت** $n \times n$ **است. مقدار** `Know[i,j]` **برابر یک است اگر فرد** `i` **فرد** `j` **را بشناسد، وگرنه صفر است.**

هدف پیدا کردن شماره‌ی `s` **است به گونه‌ای که تمامی عناصر ستون** `s`**ام (به جز** `[s,s]`**) یک و تمامی عناصر سطر** `s`**ام (به جزمولفه** `[s,s]`**) صفر باشند:**

```
Algorithm celebrity (Know);
Input:Know (an n*n Boolean matrix).
output:celebrity.
begin
   i:=1; j:=2;
   next:=3;
   {in the first phase we eliminate all but one candidate}
   while  next <= n+1 do
   if Know[i,j] then  i:=next
                else j:=next;
   next:=next+1;
   {one of either i or j is eliminated}
```

```
    if i=n+1 then  candidate:=j; else  candidate:=i;
    {Now we check that the candidate is indeed the celebrity}
    wrong:=false; k:=1;
    Know[candidate,candidate]:=false;
    {a dummy variable to pass the test}
    while not wrong and k<=n do
    if Know[candidate,k] then wrong:=true;
    if not Know[k,candidate] then
           if candidate<>k then wrong:=true;
    k:=k+1;
    if not wrong then celebrity:=candidate
                 else celebrity:=0 {no celebrity}
end;
```

مسئله: محاسبه‌ی دقیق عدد nام فیبوناچی

می‌دانیم

$$F_n = \begin{cases} ۰ & \text{if } n = ۰, \\ ۱ & \text{if } n = ۱, \ . \\ F_{n-۱} + F_{n-۲} & \text{if } n > ۱ \end{cases}$$

مثلاً $۰, ۱, ۱, ۲, ۳, ۵, ۸, ۱۳, ۲۱, ۳۴, \ldots$

هدف: محاسبه‌ی دقیق F_n **برای هر** $n \geq ۰$ **ورودی**

روش استقرایی

```
FIBONACCI(n)
 1  if n = 0
 2    then return 0
 3  if n = 1
 4    then return 1
 5  if n > 1
 6    then return FIBONACCI(n − 1)+FIBONACCI(n − 2)
```

این الگوریتم به‌اندازه‌ی مقدار F_n عمل جمع انجام می‌دهد.

می‌دانیم که

$$F_n = \langle \frac{\phi^n}{\sqrt{5}} \rangle$$

که $\langle x \rangle$ را نزدیکترین عدد صحیح به x و $\phi = (1 + \sqrt{5})/2$ نسبت طلایی (golden ration**) است.**

- **پس یک الگوریتم از $\Omega(\phi^n)$ برای این‌کار داریم.**
- **ϕ^n را می‌توان با $O(\lg n)$ بار مجذور کردن ϕ به‌دست آورد.**
- **اما، به‌علت خطای محاسبات با ممیز شناور، مقدار محاسبه‌شده برای n های بزرگ، ممکن است درست نباشد.**

روش از پایین به بالا از $\Theta(n)$

با شروع از اعداد ۰ و ۱ می‌توان با $n-1$ بار جمع عدد F_n را به‌دست آورد.

روش سریع از $\Theta(\lg n)$

لم. اعداد فیبوناچی در رابطه‌ی زیر صدق می‌کنند.

$$\begin{bmatrix} F_{n+1} & F_n \\ F_n & F_{n-1} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^n$$

اثبات. با استقراء.

برای $n = ۱$ واضح است که

$$\begin{bmatrix} F_۲ & F_۱ \\ F_۲ & F_۰ \end{bmatrix} = \begin{bmatrix} ۱ & ۱ \\ ۱ & ۰ \end{bmatrix}$$

برای $n \geq ۲$ و طبق تعریف اصلی رابطه‌های زیر برقرارند.

$$\begin{bmatrix} F_{n+۱} & F_n \\ F_n & F_{n-۱} \end{bmatrix} = \begin{bmatrix} F_n & F_{n-۱} \\ F_{n-۱} & F_{n-۲} \end{bmatrix} \cdot \begin{bmatrix} ۱ & ۱ \\ ۱ & ۰ \end{bmatrix} = \begin{bmatrix} ۱ & ۱ \\ ۱ & ۰ \end{bmatrix}^{n-۱} \cdot \begin{bmatrix} ۱ & ۱ \\ ۱ & ۰ \end{bmatrix} = \begin{bmatrix} ۱ & ۱ \\ ۱ & ۰ \end{bmatrix}^{n}.$$

پس برای به‌دست آوردن F_n باید ماتریس $\begin{bmatrix} ۱ & ۱ \\ ۱ & ۰ \end{bmatrix}$ را n بار در خودش ضرب کنیم و این کار را می‌توان در $\Theta(\lg n)$ و بدون خطای محاسباتی دقیقاً محاسبه کرد.

تحلیل میانگین در الگوریتم‌های تصادفی

مسئله‌ی استخدام

داستان: ورود برخط n نفر، روزی یک نفر، هر فرد توانایی‌ای دارد که پس از ورود مشخص می‌شود ($key[i]$).

مسئله: انتخاب فرد برتر در هر روز

هزینه‌ها:

- **بررسی هر فرد c (کم)**
- **جابه‌جایی انتخاب شده با فرد جدید m (زیاد)**

اگر همه از پیش داده شده بودند، مسئله همان یافتن عنصر بیشینه است.

اگر روزانه باید تصمیم بگیریم:

- **بیش‌ترین هزینه: اگر به ترتیب صعودی توانایی‌ها بیایند.**
- **کم‌ترین هزینه: اگر به ترتیب نزولی توانایی‌ها بیایند.**

هزینه‌ی میانگین: ورود تصادفی.

یعنی احتمال این‌که نفر iام (که پس از نفر ۱ تا $i-1$ آمده است) بین i نفر اول بهترین باشد $1/i$ است.

اگر ورودی تصادفی باشد، میانگین هزینه چقدر است؟

صورت دیگر مسئله: یافتن کوچک‌ترین دایره‌ی محاطی n نقطه‌ی ۱-بعدی

تحلیل احتمالاتی

روش ۱:

$$\begin{aligned} X_i &= I\{i\text{th person is selected}\} \\ &= \begin{cases} ۱ & \text{if person } i \text{ is selected} \\ ۰ & \text{if person } i \text{ is not selected} \end{cases} \end{aligned}$$

می‌دانیم: $E[X_i] = prob\{\text{person } i \text{ is selected}\} = ۱/i$

میانگین هزینه برابر است با n باضافه‌ی میانگین جابه‌جایی فرد انتخاب شده.

میانگین تعداد جابه‌جایی‌ها برابر میانگین متغیر تصادفی $X = X_۱ + X_۲ + \cdots + X_n$ است.

$$\begin{aligned} E[X] &= E\left[\sum_{i=۱}^{n} X_i\right] \\ &= \sum_{i=۱}^{n} E[X_i] \\ &= \sum_{i=۱}^{n} ۱/i \\ &= \ln n + O(۱) \end{aligned}$$

روش دوم: تحلیل معکوس (reverse analysis)

عناصر را به همان ترتیبی که آمده‌اند، یکی یکی بر می‌داریم.

با چه احتمالی با برداشتن عنصر i ام فرد انتخاب شده عوض می‌شود؟

پاسخ: $1/i$

پس میانگین تعداد جابه‌جایی‌ها برابر $\Sigma_{i=1}^{n} 1/i = \ln n + O(1)$ **است.**

مسئله: کوچک‌ترین دایره‌ی محاطی n نقطه

هدف: یافتن کوچک‌ترین دایره‌ی محاطی (smallest enclosing circle) n **نقطه**

کاربرد: مرکز بازوی روبات

راه‌حل کورکورانه

راه‌حل کورکورانه

$\mathcal{O}(n^۴)$

راه‌حل $\mathcal{O}(n \lg n)$

مبتنی بر ساخت دیاگرام ورونوی بر اساس دورترین فاصله (farthest point Voronoi Diagram) **که الگوریتم آن هم چندان ساده نیست.**

هدف ما

یک الگوریتم ساده و تصادفی خطی مبتنی بر استقرا

ساختار استقرایی و افزایشی راه‌حل

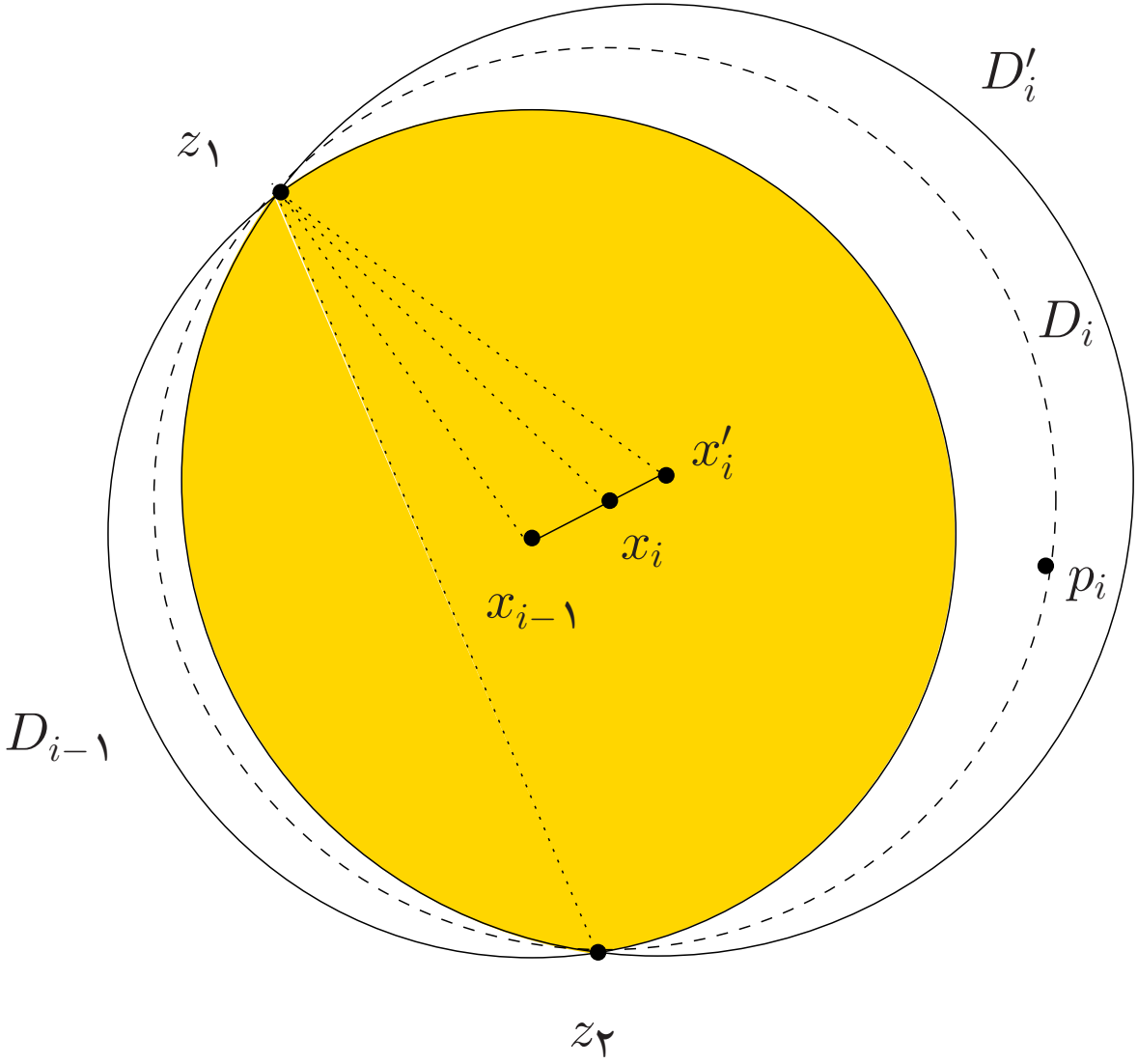
فرض: ورودی $\mathcal{P} = \{p_1, p_2, \cdots, p_n\}$ **به‌صورت تصادفی شماره‌گذاری شده است.**

فرض: $\mathcal{P}_i = \{p_1, p_2, \cdots, p_i\}$ **و** D_i **کوچک‌ترین دایره‌ی محاطی برای** $\mathcal{P}_i$

الگوریتم:

- **در مرحله‌ی i ام فرض می‌شود D_{i-1} ساخته شده است و ما نقطه‌ی p_i را اضافه می‌کنیم.**
- **اگر $p_i \in D_{i-1}$ در نتیجه $D_i = D_{i-1}$**
- **و گرنه D_i دایره‌ای است که حتماً p_i بر روی محیط آن قرار دارد.**

چرا؟


D'_i
$z_۱$
D_i
x'_i
x_i
p_i
$x_{i-۱}$
$D_{i-۱}$
$z_۲$

لم:

برای $i = ۲, \ldots, n$ **داریم:**

- **اگر** $p_i \in D_{i-۱}$، $D_i = D_{i-۱}$.
- **اگر** $p_i \notin D_{i-۱}$، p_i **بر روی محیط** D_i **است.**

الگوریتم

MiniDisc$(\mathcal{P})$
▷ Input: A set $\mathcal{P}$ of n points on the plane
▷ Output: Smallest enclosing circle for $\mathcal{P}$
1 Compute a rendom permuatation $\{p_1, \cdots, p_n\}$ of $\mathcal{P}$
2 Let D_2 be the smallest enclosing circle for $\mathcal{P}_2 = \{p_1, p_2\}$
3 **for** $i \leftarrow 3$ **to** n
4 **do if** $p_i \in D_{i-1}$
5 **then** $D_i \leftarrow D_{i-1}$
6 **else** $D_i \leftarrow$ MinDisc1Point$(\{p_1, \cdots, p_{i-1}\}, p_i)$
7 **return** D_n

MiniDisc1Point$(\mathcal{P}, q)$

▷ Input: A set $\mathcal{P}$ of n points and a point q such that,
▷ there exits an enclosing disk for $\mathcal{P}$ with q on its boundry
▷ Output: Smallest enclosing circle for $\mathcal{P}$ with q on its boundry

1 Compute a rendom permuatation $\{p_1, \cdots, p_n\}$ of $\mathcal{P}$
2 Let D_1 be the smallest circle with p_1 and q on its boundary
3 **for** $j \leftarrow 2$ **to** n
4 **do if** $p_j \in D_{j-1}$
5 **then** $D_j \leftarrow D_{j-1}$
6 **else** $D_j \leftarrow$ MinDisc2Points$(\{p_1, \cdots, p_{j-1}\}, p_j, q)$
7 **return** D_n

MiniDisc2Points$(\mathcal{P}, q_1, q_2)$

▷ Input: A set $\mathcal{P}$ of n points and two points q_1 and q_2
▷ such that, there exits an enclosing disk for $\mathcal{P}$
▷ with q_1 and q_2 on its boundry
▷ Output: Smallest enclosing circle for $\mathcal{P}$
▷ with q_1 and q_2 on its boundry

1 Compute a rendom permuatation $\{p_1, \cdots, p_n\}$ of $\mathcal{P}$
2 Let D_0 be the smallest circle with
q_1 and q_2 on its boundary
3 **for** $k \leftarrow 1$ **to** n
4 **do if** $p_k \in D_{k-1}$
5 **then** $D_k \leftarrow D_{k-1}$
6 **else** $D_k \leftarrow$ the circle on p_k, q_1, and q_2 on its border
7 **return** D_n

اثبات درستی الگوریتم

لم: فرض کنید $\mathcal{P}$ مجموعه‌ای از نقاط بر روی صفحه و R مجموعه‌ی از نقاط مجزا از $\mathcal{P}$ و $p \in \mathcal{P}$. در آن‌صورت گزاره‌های زیر برقرارند:

۱) اگر دایره‌ای هست که همه‌ی نقاط $\mathcal{P}$ را در بر بگیرد و نقاط R بر روی محیط آن باشند، در آن‌صورت کوچک‌ترین چنین دایره‌ای یکتاست و ما آن‌را به $\mathrm{md}(\mathcal{P}, R)$ نشان می‌دهیم.

۲) اگر $p \in \mathrm{md}(\mathcal{P}\backslash\{p\}, R)$ در آن‌صورت $\mathrm{md}(\mathcal{P}, R) = \mathrm{md}(\mathcal{P}\backslash\{p\}, R)$

۳) اگر $p \notin \mathrm{md}(\mathcal{P}\backslash\{p\}, R)$ در آن‌صورت $\mathrm{md}(\mathcal{P}, R) = \mathrm{md}(\mathcal{P}, R \cup \{p\})$

تحلیل الگوریتم

تحلیل معکوس

- هر بار فراخوانی MinDisc2Points به مقدار $\mathcal{O}(n)$ هزینه خواهد داشت.
- در MinDisc1Point اگر دستور شماره‌ی ۶ اجرا نشود زمان اجرای این الگوریتم همیشه $\mathcal{O}(n)$ است.
- تعیین می‌کنیم که دستور ۶ به‌طور میانگین چند بار اجرا می‌شود.
- برای این‌کار از روش «تحلیل معکوس» استفاده می‌کنیم.
  یعنی به‌جای آن‌که نقاط تک‌تک اضافه شوند آن‌ها را تک‌تک حذف می‌کنیم و احتمالی را به‌دست می‌آوریم که با حذف یک نقطه‌ی p_j دایره‌ی بهینه کوچک‌تر شود.
- این احتمال برابر است با احتمالی که دایره‌ی بهینه بزرگ‌تر شود اگر p_j اضافه شود.
  این احتمال برابر است با $\frac{۲}{j}$.

- بنابراین هزینه‌ی میانگین رویه‌ی MinDisc1Point برابر خواهد بود با

$$\sum_{j=۲}^{n} \frac{۲}{j}\mathcal{O}(j) = \mathcal{O}(n) \qquad (۱)$$

- با استفاده از این تحلیل و با همین روش به‌سادگی می‌توان دید که زمان اجرای رویه‌ی اصلی یعنی MinDisc هم به‌طور میانگین $\mathcal{O}(n)$ است.

اسلایدهای قدیمی که دیگر گفته نمی‌شوند

مسئله‌ی ۱: رنگ کردن ناحیه‌های بین خط‌ها

یک صفحه با n خط. با چند رنگ می‌توان ناحیه‌های ایجاد شده را رنگ کرد؟

جواب = ۲ رنگ

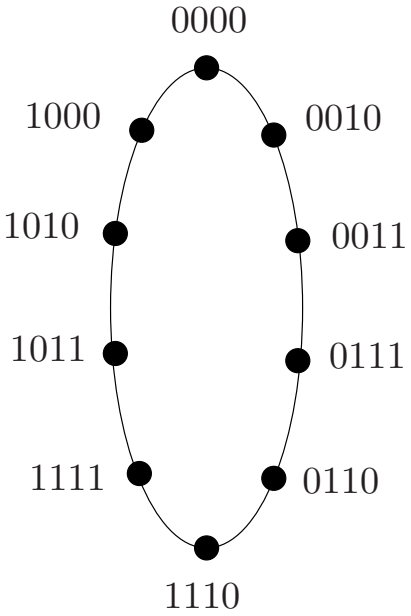
- **پایه: برای $n = ۱$**
- **فرض : برای $n - ۱$ خط**
- **حکم : خط nام را رسم می‌کنیم و ناحیه‌ها را رنگ می‌کنیم.**

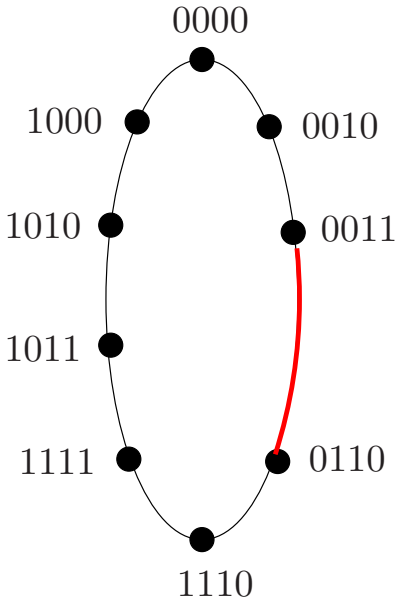
مسئله‌ی ۲: کُد گِری (Gray Code)

می‌خواهیم به n شئ کدهای متمایزی اختصاص دهیم که:

- **طول کد کمینه باشد ($\lceil \log_2 n \rceil$)**
- **کدهای متوالی (دوار) فقط در یک بیت اختلاف داشته باشند**

کدهای گری دو نوع هستند: بسته و باز.





حالت‌های ساده‌ی زیر را در نظر بگیرید:

۱) برای $n = ۲$ کد بسته‌ی بهینه داریم: $\{۱ \to ۰\}$

۲) برای $n = ۴$ نیز کد بسته‌ی بهینه داریم : $\{۱۰ \to ۱۱ \to ۰۱ \to ۰۰\}$

۳) برای $n = ۳$ کد بسته نداریم ولی کد باز بهینه داریم: $\{۱۱ \to ۰۱ \to ۰۰\}$

لم. برای تعداد زوجی عناصر ($n = ۲k$) می‌توان کد گری بسته ایجاد کرد.

اثبات استقرایی «سازنده» (constructive).

پایه‌ی استقرا: $n = ۲$ $(k = ۱)$**، بدیهی است.**

فرض استقرا: برای $n = ۲k - ۲$ **کد بسته وجود دارد.**

حکم استقرا: برای $n = ۲k$ **هم کد بسته داریم.**

لم. برای تعداد زوجی عناصر ($n = ۲k$) می‌توان کد گری بسته ایجاد کرد.

اثبات استقرایی «سازنده» (constructive).

پایه‌ی استقرا: $n = ۲$ ($k = ۱$)، بدیهی است.

فرض استقرا: برای $n = ۲k - ۲$ کد بسته وجود دارد.

حکم استقرا: برای $n = ۲k$ هم کد بسته داریم.

طول کد: $n/۲$ بیت. بد!

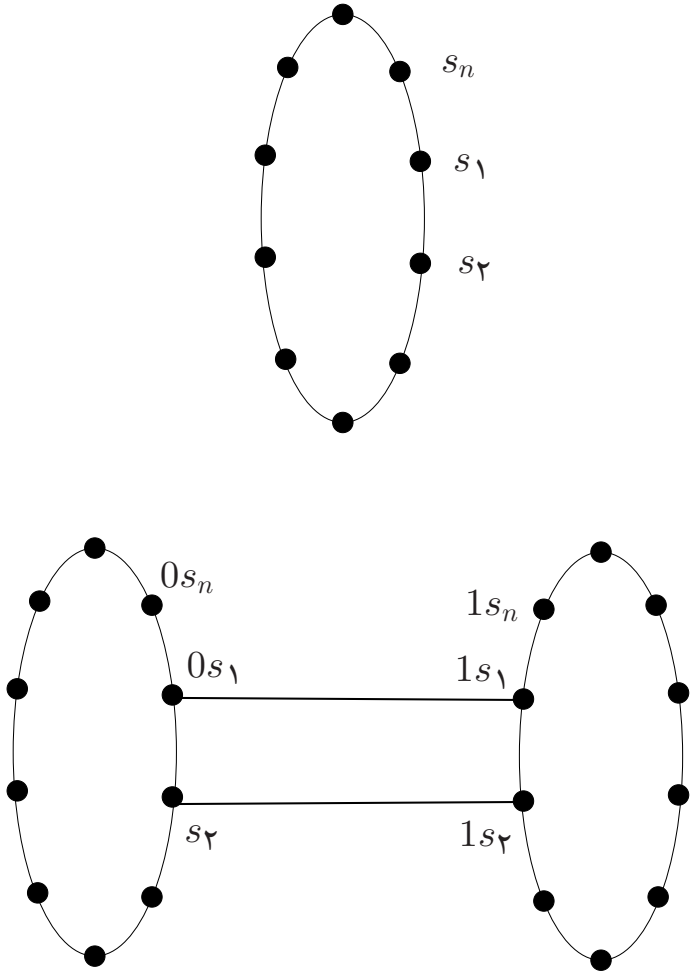
لم. برای $n = ۲^k$ عنصر می‌توان کد گریِ بسته‌ی k بیتی ایجاد نمود.

اثبات به کمک استقرا.

پایه: برای $n = ۲$ بدیهی است.

فرض: برای $n = ۲^{k-۱}$ کد بسته‌ی $k - ۱$ بیتی وجود دارد.

حکم: برای $n = ۲^k$ کد بسته‌ی k بیتی وجود دارد.


s_n
$s_۱$
$s_۲$
$0s_n$
$0s_۱$
$s_۲$
$1s_n$
$1s_۱$
$1s_۲$

لم. برای $n = ۲k + ۱$ عنصر، کد گری بسته وجود ندارد.

قضیه. برای n عدد می‌توان کد گریِ $\lceil \lg n \rceil$ بیتی ایجاد کرد. اگر n زوج باشد این کد بسته و اگر n فرد باشد باز است.

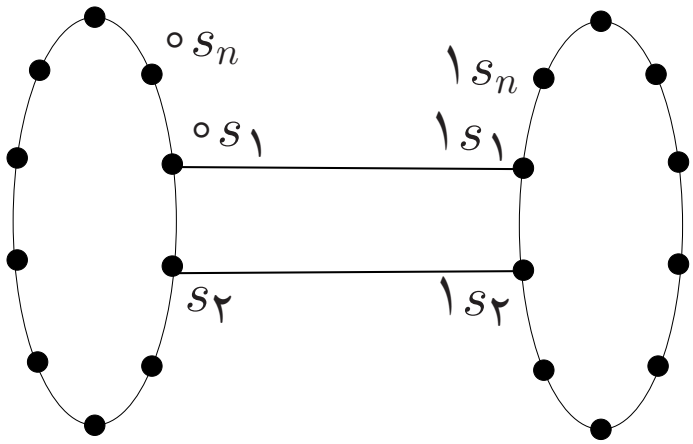
با استقرا اثبات می‌کنیم.

حالت اول: تعداد عناصر زوج است ($n = ۲k$)

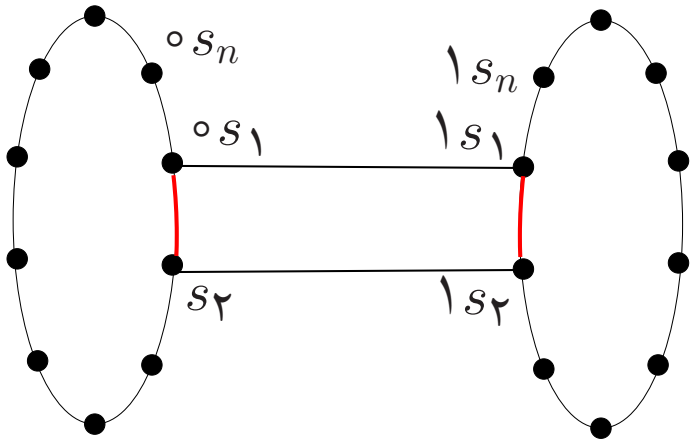
$k = \frac{n}{۲}$ زوج است یا فرد.

اگر k زوج باشد، برای k عنصر کد گری بسته، و اگر فرد باشد، کد گری باز، هر دو به طول $\lceil \lg k \rceil$ بیت وجود دارد.

برای k زوج می‌توان برای n عنصر کد گری بسته‌ی $\lceil \lg n \rceil = ۱ + \lceil \lg \frac{n}{۲} \rceil$ بیتی ایجاد کرد.



اگر k فرد باشد،



تعداد بیت‌ها:

$$\lceil \lg \frac{n}{2} \rceil + 1 = \lceil \lg n \rceil$$

حالت دوم: فرض می‌کنیم تعداد عناصر فرد است ($n = ۲k + ۱$)

برای $(۲k + ۲)$ عنصر یک کد بسته‌ی $\lceil \lg(۲k + ۲) \rceil$ بیتی ایجاد می‌کنیم.

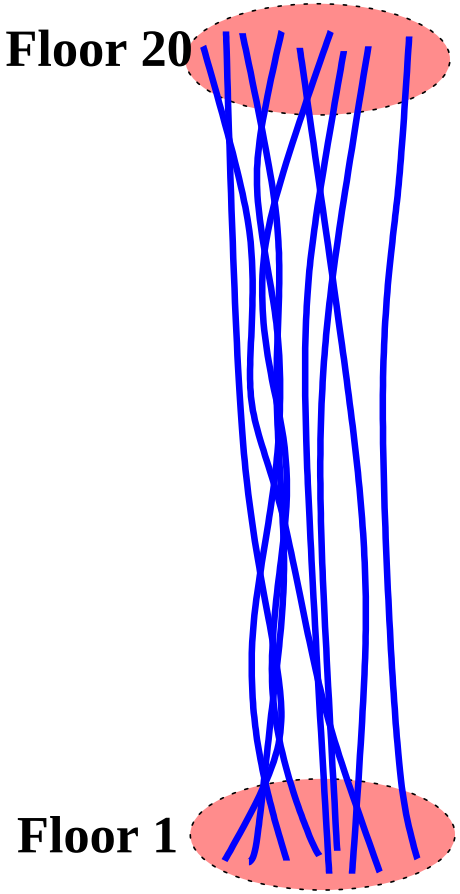
کافی است یک عنصر دلخواه را حذف کنیم

کد بسته به کد باز تبدیل و با طول بهینه تبدیل می‌شود،

چون $\lceil \lg(۲k + ۱) \rceil = \lceil \lg(۲k + ۲) \rceil$

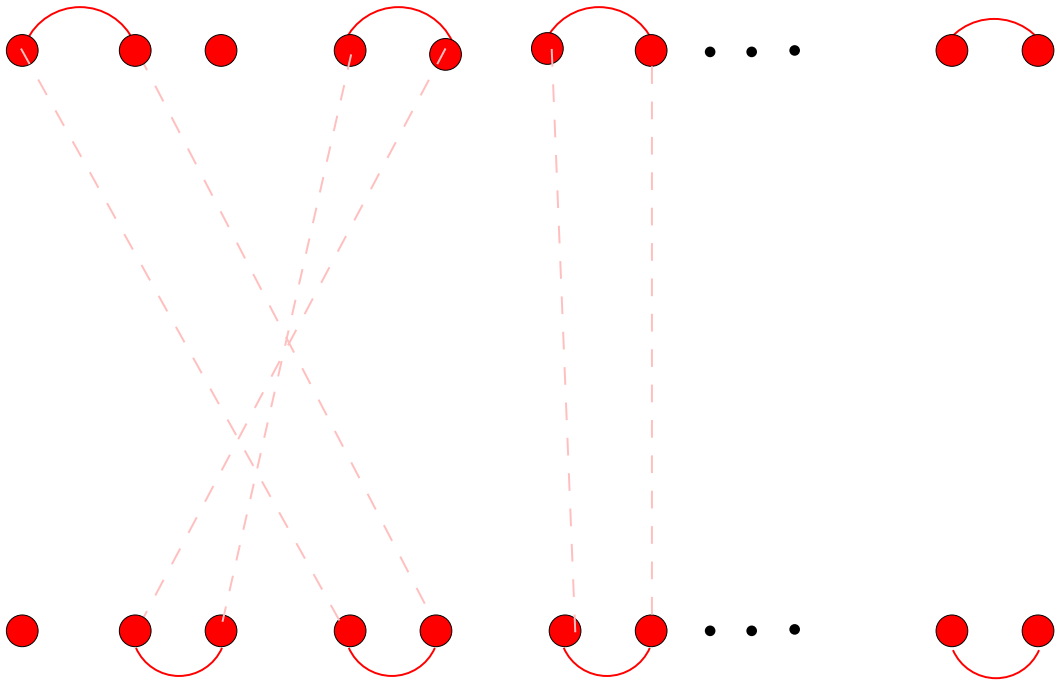
مسئله: برچسب‌گذاری سیم‌ها

- ساختمان خیلی طبقه.
- آسانسور خراب.
- می‌خواهیم با کم‌ترین «بالا و پایین» رفتن‌ها سر و ته n سیم را برچسب بزنیم تا از هم متمایز شوند.
- تنها وسیله‌ی در اختیار، دست‌گاهیی (مانند اهم‌متر) که اتصال و عدم اتصال دو سر دو سیم را بررسی می‌کند.
- و البته دفتر و مداد و یک عالمه برچسب

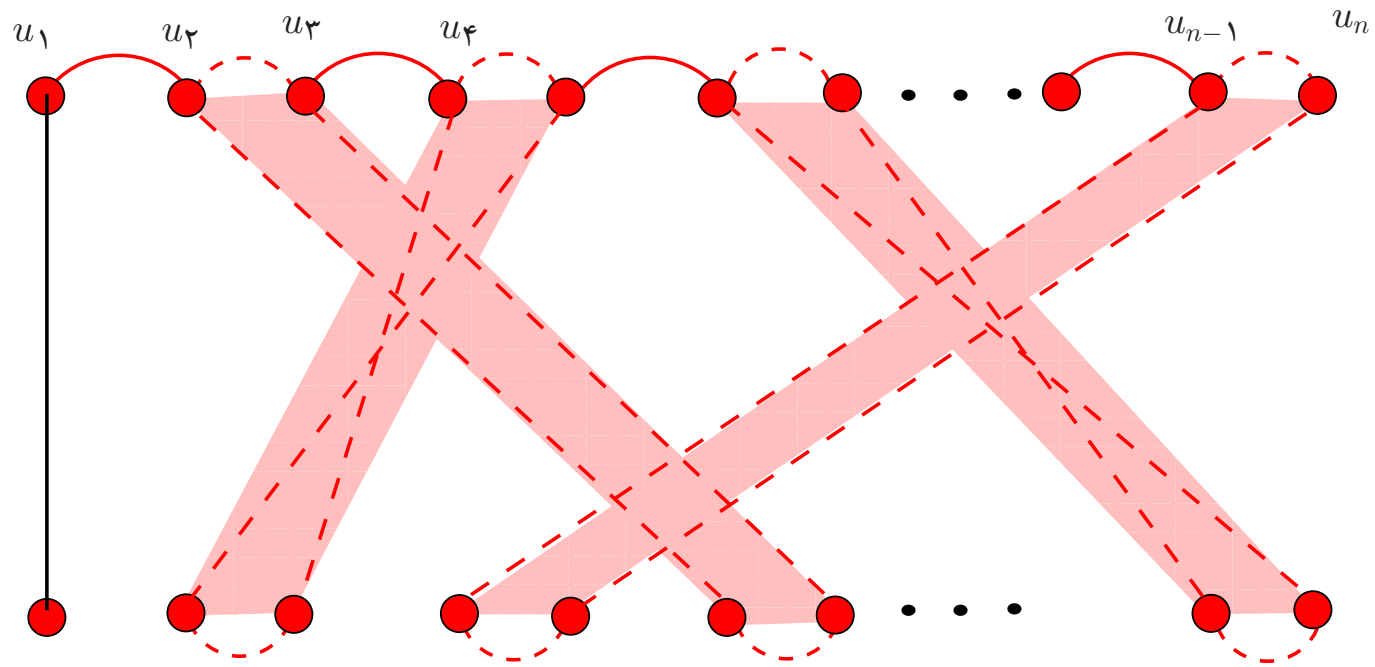

Floor 20
Floor 1

راه‌حل

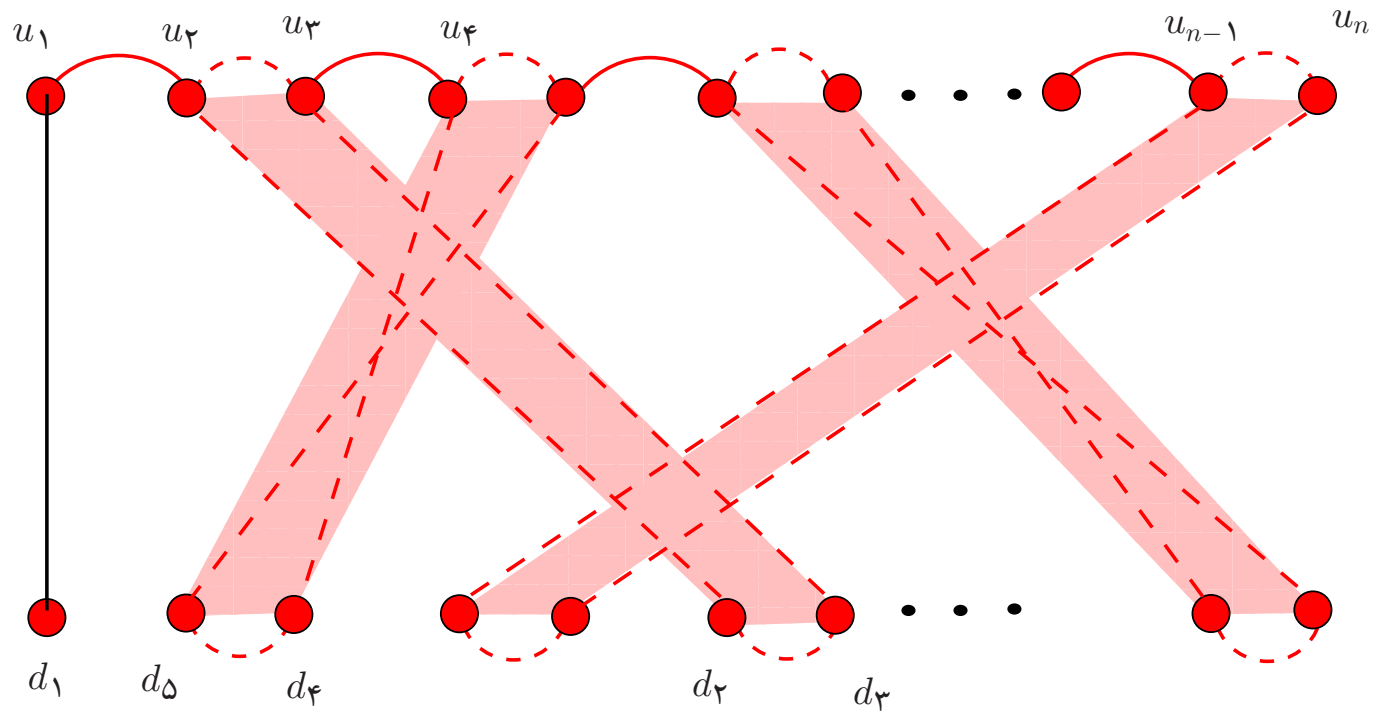
n فرد



در پایین: سیم‌ها را به دسته‌های زوج تقسیم می‌کنیم. هر زوج سیم را به‌هم وصل می‌کنیم.
در بالا: دست‌های زوج‌سیم‌ها و تک سیم مشخص می‌شود. (نمی‌دانیم که هر زوج بالایی مربوط به کدام زوج پایینی است)



در بالا: برچسب می‌زنیم. u_1 سر بالای تک سیم. بقیه مانند بالا. سر بالای سیم‌ها را مطابق شکل به هم وصل می‌کنیم.
پایین: زوج‌سیم‌ها را (با حفظ برچسب) از هم جدا می‌کنیم. با دست‌گاه به‌ترتیب سر پایین همه‌ی سیم‌ها مشخص می‌شود.


u_1
u_2
u_3
u_4
u_{n-1}
u_n
d_1
d_5
d_4
d_2
d_3

n زوج