به نام خدا

Optimal strategy for a game

استراتژی بهینه برای یک بازی

❖ مقدمه :

n سکه با ارزش های متفاوت در یک ردیف قرار گرفته اند (n زوج است). دو بازیکن به نوبت می توانند یک سکه از ابتدا یا انتهای سکه ها بردارند، تا زمانی که سکه ها تمام شود. بازیکنی که مجموع ارزش سکه هایی که برداشته، بیشتر باشد برنده بازی است.

❖ تعریف مسئله:

فرض کنید بازی را شما آغاز کنید، بیشترین ارزش سکه هایی که قطعاً می توانید به دست آورید چقدر است؟

- ✓ آیا اگر شما شروع کننده بازی باشید می توانید حتما برنده شوید؟
- ✓ آیا شروع کننده بازی بودن در نتیجه بازی تأثیر دارد؟

❖ الگوریتم بدیهی:

یک الگوریتم بدیهی برای به دست‌آوردن جواب مسئله، بررسی تمام حالت های ممکن برای بازی و سپس محاسبه حداکثر ارزش قطعی می باشد. اما به راحتی می توان دید که این الگوریتم دارای پیچدگی زمانی نمایی است. یعنی پیچیدگی از مرتبه $O(2^n)$ می باشد.

الگوریتم بدیهی به جواب می رسد ولی کارا نیست !

❖ دیگر الگوریتم ها:

با توجه به اینکه شما در هر بار دو انتخاب دارید. یکی سکه ابتدایی و یکی سکه انتهایی. یک الگوریتم می توانند به این صورت عمل کند که در هر بار سکه ای که از نظر ارزش بیشینه است را انتخاب کند.

اما این الگوریتم لزوماً برنده شدن و سود بیشینه را تضمین نمی کند !

به عنوان مثال فرض کنید ۴ سکه به صورت زیر داشته باشیم:

5 , 8 , 3 , 2

این الگوریتم اولین انتخاب شما را ۵ انتخاب می کند و حریف شما ۸ را می تواند انتخاب کند و در نهایت شما می توانید ۳ را انتخاب و سکه ۲ برای حریف شما بماند. که مجموع سود حریف بیش از شماست. !

❖ یک الگوریتم حریصانه:

در ابتدا مجموع سکه های با اندیس فرد را محاسبه و A می نامیم. سپس مجموع سکه ها با اندیس زوج را محاسبه و B می نامیم. اگر A>B آنگاه سکه اول را انتخاب و در هر مرحله سکه با اندیس فرد را انتخاب کنید اگر B>A آنگاه سکه n را انتخاب و در هر مرحله سکه با اندیس زوج را انتخاب کنید.

- ✓ چرا اگر ابتدا اندیس زوج (فرد) را انتخاب کنید می توانید تمام سکه های با اندیس زوج (فرد) را انتخاب کنید؟

این الگوریتم برنده شدن را تضمین می کند ولی سود بیشینه را تضمین نمی کند !

به عنوان مثال فرض کنید ۶ سکه به صورت زیر داشته باشیم:

3 , 2 , 2 , 3 , 1 , 2

در نتیجه :

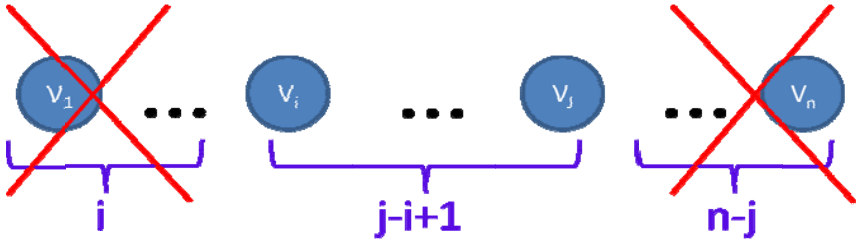
A= 3+2+1 = 6 , B = 2+3+2 = 7

اما اگر ابتدا شما سکه۳ سمت چپ را انتخاب کنید، حریف شما حتماً سکه با ارزش ۲ (از سمت چپ یا راست) خواهد برداشت. شما مجددا سکه با ارزش ۲ بردارید. در این وضعیت سکه های 1 , 3 , 2 باقی خواهند ماند. بهترین انتخاب برای حریف شما سکه با ارزش ۲ هست و در نهایت سکه با ارزش ۳ را شما برداشته و سکه ۱ را حریفتان. در نتیجه سود قطعی که به دست آمد ۸ بود که از B=۷ بیشتر است !!

❖ راه حل پویا :

۱) بررسی خاصیت زیر ساختار بهینه

فرض کنید جواب بهینه برای مسئله را داشته باشیم، فرض کنید بعد از اجرای چند مرحله i سکه از ابتدای سکه ها و n-j سکه از انتها برداشته شود یعنی سکه های با اندیس بین i و j باقی مانده و نوبت شما برای برداشتن سکه باشد. $(0\le i,j<n)$



ادعا این است که زیر جواب ، برای سکه های باقی مانده باید جواب بهینه باشد. اثبات با استفاده از تکنیک ببر و بچسبان (cut&paste) قابل انجام است.

۲) راه حل بازگشتی :

فرض کنید سکه ها را به ترتیب از ابتدا با $A_1 , A_2 , \dots , A_n$ نمایش دهیم آنگاه تعریف می کنیم :

P(i,j)= حداکثر سود قطعی که از سکه های A_i تا A_j می توان بدست آورد با این فرض که نوبت شما باشد.

V(i)= ارزش سکه iام

برای سادگی محاسبه p(i,j) فرض شرط زوج بودن تعداد سکه های باقیمانده را در نظر نمی‌گیریم و مسئله را در حالت کلی حل میکنیم بنابراین فرض کنید شما A_iرا انتخاب کنید آنگاه حریف شما یا A_{i+1} را انتخاب می کند یا A_j را. اگر A_{i+1} را انتخاب کند آنگاه سود شما v(i) + p(i+2,j) خواهد بود و اگر A_j را انتخاب کند سود شما برابر v(i) + p(i+1,j-1) خواهد بود.

اما شما نمی دانید حریفتان کدام سکه را انتخاب خواهد کرد فرض کنید بین p(i+1,j-1) و p(i+2,j) هر کدام بیشتر باشد آنرا حریفتان انتخاب کند. لذا سود قطعی شما اگر A_i را انتخاب کنید برابر

P1= Min{ p(i+2,j) , p(i+1 , j-1) } + v(i)

خواهد بود. به همین ترتیب اگر شما A_j را انتخاب کنید سود قطعی شما برابر

P2= Min{ p(i+1,j-1) , p(i , j-2) } + v(j)

خواهد بود ، لذا شما بین A_i و A_j سکه ای را انتخاب می کنید که سود قطعی حاصل از آن ماکزیمم باشد یعنی :

P(i,j) = max { P1 , P2 }

اگر رابطه بازگشتی را به صورت کلی بنویسیم داریم :

$$P(i,j)=\begin{cases} V(i) & ; \ i=j \\ \mathbf{Max\{}\mathrm{Min}\{ p(i+1,j-1) , p(i , j-2)\} + v(j) \ \mathbf{,}\mathrm{Min}\{ p(i+2,j) , p(i+1 , j-1)\} + v(i)\mathbf{\}} & \\ \mathbf{0} & ; \ \text{o.w} \end{cases}$$

۳) راه حل پویا :

هر چند می توان با یک الگوریتم بازگشتی این رابطه را پیاده سازی کرد اما زمان اجرای این الگوریتم نمایی نسبت به n خواهد بود. (چرا؟) با استفاده از تکنیک به خاطر سپاری می توان این زمان اجرا را بهبود داد ولی در اینجا با استفاده از روش برنامه ریزی پویا و از پایین به بالا مسئله را حل می کنیم.

با در نظر گرفتن یک جدول n*n و محاسبه مقادیر آن می توان p(1,n) که جواب بهینه برای مسئله است را محاسبه کرد. با توجه به اینکه p(i,j) به مقادیر p(i+1,j-1),p(i+2,j),p(i,j-2) بستگی دارد می توانیم جدول را از روی قطر اصلی به صورت قطری پر کنیم. (همانند ضرب زنجیر ماتریسها)

در ابتدا کلیه مقادیر جدول P را برابر صفر قرار می دهیم سپس با استفاده از الگوریتم زیر جواب مسئله را می یابیم :

```
Optimal strategy for a game ( n ,v[] )
for i=1 to n do
    p(i,i)=v(i);
end
for  k=2 to n do
    for i=1 to n-k+1 do
        j=i+k-1;
        p1 = Min{ p(i+2,j) , p(i+1, j-1)} + v(i);
        p2 = Min{ p(i+1,j-1) , p(i , j-2)} + v(j);
        if p1>p2  then
            p(i,j) = p1;
            m(i,j) = i;
        else
            P(i,j)=p2;
            m(i,j)=j;
        end
    end
end
return p,m
```

برای محاسبه پیچیدگی زمانی این الگوریتم یک حلقه در ابتدای برنامه قطر اصلی را مقدار دهی می کند که از O(n) هست ، سپس دو حلقه تو در تو برای محاسبه مقادیر داریم فرض می کنیم محاسبات داخل حلقه عدد ثابت c باشد بنابراین :

$$\sum_{k=2}^{n}\sum_{i=1}^{n-k+1} c = \sum_{k=2}^{n}(n-k+1)c = (n+1)(n-1) - \sum_{k=2}^{n} k = n^2 - 1 - \frac{n(n+1)}{2} + 1$$

$$= n^2 - \frac{n^2+n}{2}$$

بنابراین پیچیدگی زمانی این الگوریتم $O(n^2)$ است.

۴) ساخت جواب بهینه :

برای ساخت جواب بهینه از ماتریس کمکی m(n*n) استفاده می کنیم. و متناسب با هر مقدار p(i,j) اگر p1>p2 بود مقدار" i" و اگر p2>p1 مقدار "j" را در m(i,j) ذخیره می کنیم. در هر مرحله از بازی برای انتخاب بین سکه‌های A_i و A_j به m(i,j) مراجعه می کنیم اگر مقدار i در آن ذخیره شده بود سکه A_i و اگر j ذخیره شده بود سکه A_j را انتخاب می کنیم. به این طریق به جواب بهینه دست خواهیم یافت. توجه کنید که ساخت جواب بهینه وابسته به انتخاب حریف است و ممکن است از p(1,n) بیشتر شود ولی کمتر نخواهد شد.

مثال :

فرض کنید ۶ سکه مطابق زیر داشته باشیم :

7 , 9 , 2 , 6 , 4 , 3

	۱	۲	۳	۴	۵	۶
۱	7	9 j	8 i	15 j	12 i	18 j
۲		9	9 i	10 i	15 i	14 i
۳			1	6 j	5 i	9 j
۴				6	6 i	9 i
۵					4	4 i
۶						3

برای سادگی محاسبه p(i,j) فرض زوج بودن تعداد سکه ها را در نظر نگرفتیم، شما میتوانید جهت بهبود الگوریتم این شرط را در نظر گرفته و محاسبات اضافی را حذف کنید و قطرهای ماتریس را یک در میان پر کنید. اما چون پیچیدگی به اندازه یک عدد ثابت بهبود می یابد از نظر تئوری چندان فرقی نمیکند.