فرض کنید جند جمله ای مشخصه رابطه بازگشتی بصورت زیر باشد

$$a_n = c_1 a_{n-1} + c_2 a_{n-2}$$

$r^2 - c_1 r - c_2 = 0$

Theorem 1.4. *If $r_1 \neq r_2$, then there are constants α_1 and α_2 such that*

$$a_n = \alpha_1 r_1^n + \alpha_2 r_2^n$$

for all $n \geq 0$.

$$\begin{bmatrix} c_1 & c_2 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} a_{n-1} \\ a_{n-2} \end{bmatrix} = \begin{bmatrix} a_n \\ a_{n-1} \end{bmatrix} \longleftrightarrow \begin{aligned} c_1 a_{n-1} + c_2 a_{n-2} &= a_n \\ a_{n-1} &= a_{n-1} \end{aligned}$$

1.2.2. *Using matrices.* Our recurrence relation translates to the following matrix equation:

$$\begin{bmatrix} c_1 & c_2 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} a_{n-1} \\ a_{n-2} \end{bmatrix} = \begin{bmatrix} a_n \\ a_{n-1} \end{bmatrix} \Rightarrow \begin{bmatrix} a_n \\ a_{n-1} \end{bmatrix} = C \begin{bmatrix} a_{n-1} \\ a_{n-2} \end{bmatrix}$$

$C^2 = CC = BD\underbrace{B^{-1}B}_{I}DB^{-1} = BD^2B^{-1}$

for $n \geq 2$. Set $C = \begin{bmatrix} c_1 & c_2 \\ 1 & 0 \end{bmatrix}$. Then our goal is to find a formula for $C^n \begin{bmatrix} a_1 \\ a_0 \end{bmatrix}$ for all $n \geq 0$ because the second entry is a_n. Thinking back to linear algebra, we can do this if we can diagonalize C: if $C = BDB^{-1}$ for some diagonal matrix D, then we have $C^n = BD^nB^{-1}$ and D^n is easy to compute. The characteristic polynomial of C is conveniently $t^2 - c_1 t - c_2$, which is the characteristic polynomial of the recurrence relation, so its eigenvalues are r_1, r_2. Since we are assuming they are distinct, C is diagonalizable, so there is some matrix B such that $C = B \begin{bmatrix} r_1 & 0 \\ 0 & r_2 \end{bmatrix} B^{-1}$.

Let's just name $\begin{bmatrix} x \\ y \end{bmatrix} = B^{-1} \begin{bmatrix} a_1 \\ a_0 \end{bmatrix}$. We don't need to compute x, y but note that they are constants of our recurrence relation (they do not depend on n). Then

$$\begin{bmatrix} a_{n+1} \\ a_n \end{bmatrix} = C^n \begin{bmatrix} a_1 \\ a_0 \end{bmatrix} = B \begin{bmatrix} r_1^n & 0 \\ 0 & r_2^n \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} b_{1,1} x r_1^n + b_{1,2} y r_2^n \\ b_{2,1} x r_1^n + b_{2,2} y r_2^n \end{bmatrix}.$$

To finish the proof we just set $\alpha_1 = b_{2,1} x$ and $\alpha_2 = b_{2,2} y$.

$= BD^n \underbrace{B^{-1} \begin{bmatrix} a_1 \\ a_0 \end{bmatrix}}_{\begin{bmatrix} x \\ y \end{bmatrix}}$

$a_n = b_{21} x r_1^n + b_{22} y r_2^n$

مثال 1:

$$a_n - 3a_{n-1} + 2a_{n-2} = \underline{n4^n}$$

$f(n)$ $\qquad s = 4$

نخست همگن $\quad r^2 - 3r + 2 = 0 \Leftarrow (r-2)(r-1) = 0$

$r = 2, \ r = 1$

$a_n^{(h)} = \alpha_1 1^n + \alpha_2 2^n = \alpha_1 + \alpha_2 2^n$

$a_n^{(p)} = (\beta_1 n + \beta_2) 4^n$

$a_n^{(p)} - 3a_{n-1}^{(p)} + 2a_{n-2}^{(p)} = n4^n$

$(\beta_1 n + \beta_2)4^n - 3(\beta_1(n-1) + \beta_2)4^{n-1} + 2(\beta_1(n-2) + \beta_2)4^{n-2} = n4^n$

$16\beta_1 n + 16\beta_2 - 12\beta_1 n + 12 - 12\beta_2 + 2\beta_1 n - 4 + 2\beta_2 = 16n$

$\forall n \quad (6\beta_1 - 16)n + (6\beta_2 + 8) = 0 \qquad \beta_1 = \frac{16}{6} = \frac{8}{3} \qquad \beta_2 = -\frac{8}{6}$

جواب کلی

$a_n = (\frac{8}{3}n - \frac{8}{6}) 4^n + \alpha_1 + \alpha_2 2^n$

با شرایط اولیه α_1 و α_2 بدست می‌آید

برخی معادلات بازگشتی مانند مساله برج هانوی بصورت زیر هستند:

$$T(n) = \cancel{a}T(n-b) + c = T(n-2b) + c + c$$

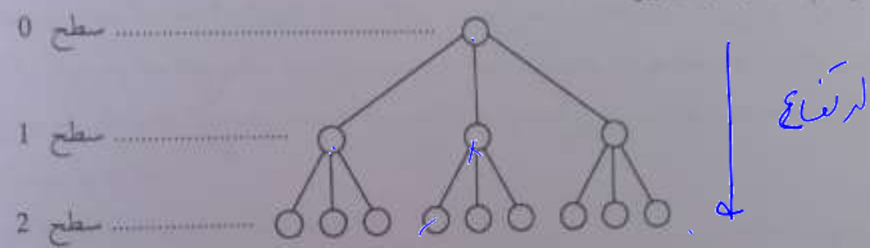
$= \dots = T(0) + c + \dots + c$

قضیه ۱: در حالت کلی می‌توان اثبات کرد ($a > 0$، $b > 0$ و $c > 0$ اعداد ثابت هستند):

$$T(n) = aT(n-b) + c \Rightarrow \begin{cases} T(n) = \theta\left(a^{\frac{n}{b}}\right) & : \ \underline{a \neq 1} \ \text{اگر} \\ T(n) = \theta\left(\frac{n}{b}\right) = \underline{\theta(n)} & : \ \underline{a = 1} \ \text{اگر} \end{cases}$$

توجیه فرمول فوق به کمک درخت بازگشتی ساده است. اگر $a = 1$ باشد یعنی هر گره درخت فقط یک فرزند دارد و تعداد سطوح درخت هم $\theta\left(\frac{n}{b}\right)$ می‌شود پس در این حالت درخت در کل $\theta\left(\frac{n}{b}\right) = \theta(n)$ گره خواهد داشت.

حال برای حالت $a \neq 1$ فرمول تعداد کل گره‌های درخت (مربوط به درس ساختمان داده‌ها) را یادآوری می‌کنیم. مثلاً در درخت زیر:



تعداد شاخه شدن $a=3$ ، ارتفاع $= h =$ تعداد سطوح $= 2$ ، تعداد کل گره‌ها $= 3^0+3^1+3^2$

و در حالت کلی:

تعداد کل گره‌ها $= 1+a^1+a^2+\cdots+a^h$ (جمع تصاعد هندسی $= \frac{t_1(q^n-1)}{q-1}$)

تعداد کل گره‌ها $= \frac{1\times(a^{h+1}-1)}{a-1} \Rightarrow$ $\boxed{\text{تعداد کل گره‌های درخت } a \text{ شاخه به ارتفاع } h = \theta(a^h)}$

در درخت بازگشتی $T(n)=aT(n-b)+c$ تعداد سطوح $h=\theta\left(\frac{n}{b}\right)$ است پس:

$T(n)$ تعداد کل گره‌های درخت $= \theta(a^h) = \theta(a^{\frac{n}{b}})$

مثال 2:

مثال ۲۶:

$T(n)=3T(n-2)+5 \Rightarrow T(n)=\theta(3^{\frac{n}{2}})$ a=۳ b=۲

$T(n)=T(n-1)+1 \Rightarrow T(n)=\theta(n)$ a=1, b=1 — مسأله فاکتوریل

$T(n)=2T(n-1)+1 \Rightarrow T(n)=\theta(2^n)$ — مسأله برج‌های هانوی

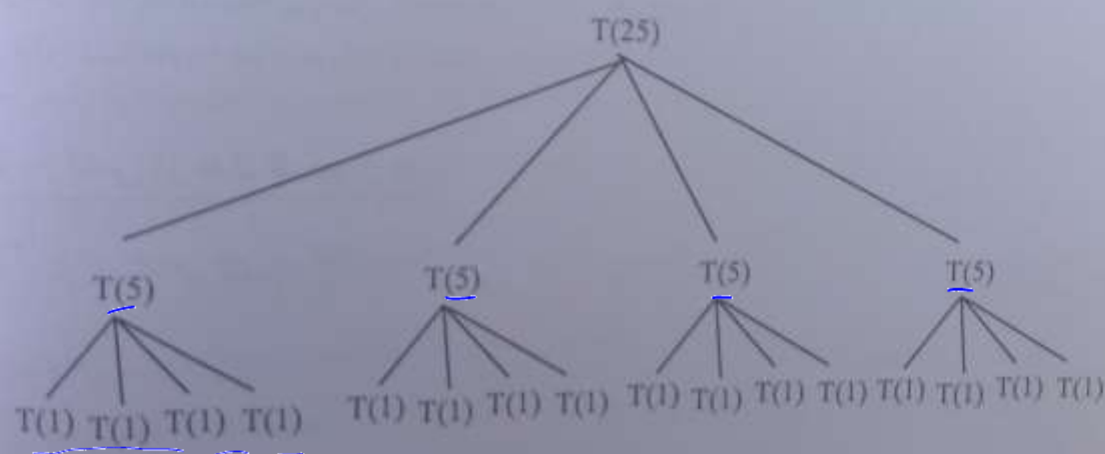
$T(n)=2T(n-2)+1 \Rightarrow T(n)=\theta(2^{\frac{n}{2}})$

پیچیدگی زمانی برخی از الگوریتم ها بصورت زیر است:

پیچیدگی زمانی بسیاری از

$$\begin{cases} T(n) = aT(\frac{n}{b}) \\ T(1) = 1 \end{cases}$$

مثلاً برای $a = 4$ و $b = 5$ و $n = 25$ تعداد صدا زدن‌ها به شکل زیر است :



ارتفاع درخت شکل فوق برابر ۲ است یعنی $\log_5^{25} = 2$ که در حالت کلی $\log_b^n$ می‌شود. در هر سطح هر گره به 4 گره تقسیم می‌شود که در حالت کلی هر گره به a انشعاب تقسیم می‌گردد. پس :

تعداد کل صدا زدن‌ها در درخت درجه a به ارتفاع h $= 1 + a + a + a^2 + ... + a^h$

$$= \frac{1 \times (a^{h+1} - 1)}{a - 1} \qquad (a \neq 1)$$

توجه کنید که h را برابر $\log_b^n$ در نظر گرفته‌ایم و ارتفاع، تعداد سطوح منهای یک است.

$$\text{تعداد کل صدازدن‌ها} = \frac{a^{h+1}-1}{a-1} = \frac{a^{1+\log_b^n}-1}{a-1}$$

مثلاً در شکل فوق که $n = 25$ ، $b = 5$ و $a = 4$ است:

$$\text{تعداد کل صدا زدن‌ها} = \frac{4^{1+\log_5^{25}}-1}{4-1} = \frac{4^3-1}{3} = \frac{63}{3} = 21$$

در حالت کلی:

$$\text{تعداد صدا زدن‌ها} = \frac{a \times a^{\log_b^n}-1}{a-1} = \theta(a^{\log_b^n}) = \theta(n^{\log_b^a})$$

پس در حالت کلی داریم (c,b,a اعداد مثبت ثابتی هستند و b>1 و $a \geq 1$):

$$\begin{cases} T(n) = aT(\frac{n}{b}) + c \\ T(1) = 1 \end{cases} \Rightarrow \begin{cases} \theta\left(a^{\log_b^n}\right) = \theta(n^{\log_b^a}) & (a \neq 1) \\ \theta(\log_b^n) & (a = 1) \end{cases}$$

$h = \log_b^n$

$$\begin{cases} a^{\log_b^n} = n^{\log_b^a} \\ \log_b^n \log_b^a = \log_b^a \log_b^n \end{cases}$$

$c_1 a^h \leq \frac{a^{h+1}-1}{a-1} < c_2 a^h$

$a^{h+1}-1 < c_2 a^{h+1} - c_2 a^h$

$c_2 a^h - 1 < (c_2-1)a^{h+1}$

$c_2 a^h < (c_2-1)a^{h+1}$

$c_2 < (c_2-1)a$

$\frac{c_2}{c_2-1} < a$

مثال 3:

$T(n) = T(\frac{n}{2}) \Rightarrow \theta(\log_2 n)$

$T(n) = 3T(\frac{n}{5}) + 7 \Rightarrow \theta\left(n^{\log_5 3}\right)$

$T(n) = 2T(\frac{n}{2}) + 1 \Rightarrow \theta\left(n^{\log_2^2}\right) = \theta(n)$

مثال 4:

Gcd

```
int  BMM (int a, int b)     (a > b > 0)
 {
    if (b==0)  return a;
     else return BMM (b, a mod b);
  }
```

$m_0 \quad m_1 \quad m_2$

۳۰ = ۲۶ × ۱ + ۴

۲۶ = ۴ × ۶ + ۲

۴ = ۲ × ۲ + ۰

مثلاً داریم : $BMM(30,26) = BMM(26, 4) = BMM(4, 2) = BMM(2, 0) = 2$

اثبات : دنباله اعداد تولید شده توسط فراخوانی تابع فوق از چپ به راست به صورت زیر است :

$m_1, m_2, \ldots, m_{i-1}, m_i, m_{i+1}, \ldots, 0$

که در دنباله فوق $m_1 = a$ و $m_2 = b$ و $m_{i+1} = m_{i-1} \bmod m_i$ است. بدیهی است برای حالت $a > b > 3$ داریم $a \bmod b < \frac{a}{2}$.

$a = bk + r$

$r = a \bmod b$

$r < \frac{a}{2}$?

$a - bk < \frac{a}{2}$

$\frac{bk+r}{2} < bk \iff \frac{a}{2} < bk$

$r < bk$

به همین ترتیب در دنباله فوق داریم $\frac{m_{i-1}}{2} > \underbrace{m_{i-1} \bmod m_i}_{m_{i+1}}$ می‌باشد. پس :

$$m_{i+1} = m_{i-1} \bmod m_i < \frac{m_{i-1}}{2} \Rightarrow m_{i+1} < \frac{m_{i-1}}{2}$$

یعنی در بدترین شرایط در هر بار فراخوانی بازگشتی، پیچیدگی نصف می‌شود لذا مرتبه الگوریتم مذکور $O(\log_2 a)$ است که بهتر است این نکته را حفظ کنید.

دسته ای دیگر از معادلات بازگشتی که در تحلیل الگوریتم هایی مانند جستجوی ادغامی یا یافتن بزرگترین عنصر در یک آرایه پدید می آیند بصورت زیر هستند:

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

مثال 5: هدف یافتن بزرگترین عنصر یک آرایه است:

```
Algorithm FindMax ( A, low, high, fmax ),
if (low = high) then fmax = A[low]
    else if (high = low + ۱) then
        if A[low] > A[high] then
            fmax = A[low]
        else fmax = A[high]
    else
    {
        mid = ⌊(low + high) / ۲⌋
        FinMax (A , low , mid , lmax);
        FindMax (A , mid +۱ , high, rmax);
        if lmax > rmax then fmax = lmax
            else fmax = rmax
    }
```

$$\begin{cases} t(n) = ۲t\left(\frac{n}{۲}\right) + ۱ \ , \quad n > ۲ \\ t(۱) = ۰ \ , \quad t(۲) = ۱ \end{cases}$$

$$t(n)=۱+۲t\left(\frac{n}{۲}\right)=۱+۲\left(۱+۲t\left(\frac{n}{۲^۲}\right)\right)=۱+۲+۲^۲t\left(\frac{n}{۲^۲}\right)$$

$$=۱+۲+۲^۲\left[۱+۲t\left(\frac{n}{۲^۳}\right)\right]=۱+۲+۲^۲+۲^۳t\left(\frac{n}{۲^۳}\right)$$

$$=۱+۲+۲^۲+۲^۳+\cdots+۲^{m-۱}+۲^{m}\overset{۱}{t(۱)}$$

$$=۱+۲+۲^۲+۲^۳+\cdots+۲^{m-۱} = \frac{۲^m-۱}{۲-۱}$$

$$=۲^{m}-۱=n-۱=\theta(n)$$

$n = ۲^m$

Theorem 4.1 (Master theorem)
Let $a \geq 1$ and $b > 1$ be constants, let $f(n)$ be a function, and let $T(n)$ be defined on the nonnegative integers by the recurrence

$T(n) = aT(n/b) + f(n)$,

where we interpret n/b to mean either $\lfloor n/b \rfloor$ or $\lceil n/b \rceil$. Then $T(n)$ has the following asymptotic bounds:

1. If $f(n) = O(n^{\log_b a - \epsilon})$ for some constant $\epsilon > 0$, then $T(n) = \Theta(n^{\log_b a})$.
2. If $f(n) = \Theta(n^{\log_b a})$, then $T(n) = \Theta(n^{\log_b a} \lg n)$.
3. If $f(n) = \Omega(n^{\log_b a + \epsilon})$ for some constant $\epsilon > 0$, and if $af(n/b) \leq cf(n)$ for some constant $c < 1$ and all sufficiently large n, then $T(n) = \Theta(f(n))$. ■

مثال 6: $T(n) = 2T\left(\frac{n}{2}\right) + 1$

$T(n) = \theta(n)$ حالت ۱ $a = ۲$ $b = ۲$ $\log_b a = 1$ $f(n) = 1$ $\epsilon = 1/2$

$n^{\log_b a} = n^1 \rightarrow f(n) \leq O(n^{1-\epsilon})$

$O(\sqrt{n})$

مثال 7: $T(n) = 2T\left(\frac{n}{2}\right) + n$

$\theta(n \log n)$ حالت ۲ $a = ۲$ $b = ۲,$ $f(n) = n$

$n^{\log_b a} = n$ $f(n) = \theta(n^{\log_b a}) = \theta(n)$

مثال 8: $T(n) = 2T\left(\frac{n}{3}\right) + 1$

$a = 2 \quad b = 3 \quad \log_b^a = \log_3^2 > 0$

$f(n) = 1 = O(n^{\log_3^2 - \varepsilon}) \quad \varepsilon = \frac{1}{2}\log_3^2$

حالت الف

$\Theta(n^{\log_3^2})$

مثال 9:

$T(n) = 8T\left(\frac{n}{9}\right) + \text{n}\log n$

$a = 8 \quad b = 9 \quad n^{\log_b^a} = n^{\log_9^8} < n^1$

$n\log n > C\, n^{\log_9^8 + \varepsilon}$? $\quad \log_9^8 + \varepsilon < 1$ $\quad \varepsilon$ کوچک

$\Theta(n\log n)$ حالت ۳

مثال 10:

$T(n) = 4T\left(\frac{n}{2}\right) + 5\,n^2 + \text{n}\log n + 6n$

$\Theta(n^2) \Rightarrow$ حالت ۲

$a = 4 \quad b = 2 \quad n^{\log_b^a} = n^2$

$\Theta(n^2 \log n)$

مثال 11: حل به کمک تغییر متغیر

$T(n) = 2T(\sqrt{n}) + \log n$

$n = 2^m \quad \sqrt{n} = n^{\frac{1}{2}} = (2^m)^{\frac{1}{2}} = 2^{\frac{m}{2}}$

$T(2^m) = 2T(2^{\frac{m}{2}}) + m$

تعریف $S(m) = T(2^m)$

$S(m) = 2S(\frac{m}{2}) + m$

$a = 2 \quad b = 2 \quad m^{\log_b^a} = m \quad f(m) = m = \Theta(m)$

$m = \log n \Leftarrow \Theta(m\log m)$: حالت ۲

چند تست کنکور 98

۷۰- مرتبه زمانی الگوریتم زیر کدام است؟

```
f(n)
{
   x = ۲; y = ۱;
   while(y <= n){
         y = y × x;
         x = x × x;
     }
}
```

۱) $n\log n$ ۲) $\log n \log n$ ۳) $\log(\log n)$ ۴) $\log(\log(\log n))$

۷۲- فرض کنید زیر برنامه $C = aux(A,B)$ دو ماتریس A و B با اندازه $n \times n$ را ضرب کرده و نتیجه را در C بر می‌گرداند. مقدار برگشتی تابع زیر برای ماتریس M کدام است؟

```
Mat(M)
{
  P = M ; Q = M
  for i = ۱  to n − ۱ do{
      P = aux(P,M)
      Q = aux(Q,P)
  }
  return(Q)
}
```

۱) M^n ۲) M^{n-1} ۳) $M^{\frac{n(n-1)}{2}}$ ۴) $M^{\frac{n(n+1)}{2}}$

۷۹- اگر $T(n) = 2T(\frac{n}{4}) + \theta(1)$ از مرتبه $\theta(\sqrt{n})$ باشد، آنگاه $T(n) = 2T(\frac{n}{4}) + \theta(\sqrt{n})$ از کدام مرتبه است؟

۱) $O(n)$ ۲) $O(\log^2 n)$ ۳) $O(n\sqrt{\log n})$ ۴) $\theta(\sqrt{n}\log n)$

۸۰- اگر $U(m) = T(2^m)$ باشد و $T(n) = T(\frac{n}{2}) + \log n$، آنگاه رابطه بازگشتی U کدام است؟

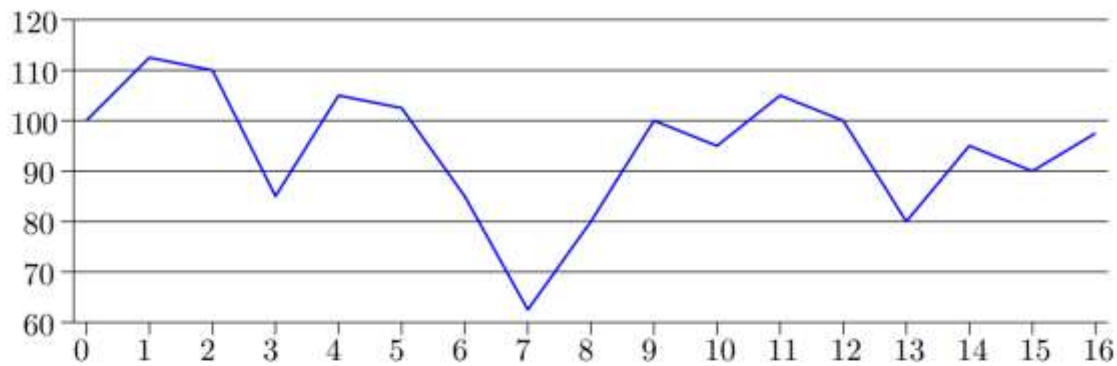
۱) $U(m) = U(m-1) + 1$ ۲) $U(m) = U(m-1) + m$

۳) $U(m) = U(m-1) + \log m$ ۴) $U(m) = U(m-1) + 2^m$

Maximum Subarray Problem

Example: I am giving you perfect stock market predictions of a company ABC for the next thirty days. But under a condition — you can buy and sell the stocks only once. That's it. You can't do more than one "buy-sell" transaction. What would be the best day to buy the stocks? And when do you need to sell them to make the maximum profit?

Buying and Selling



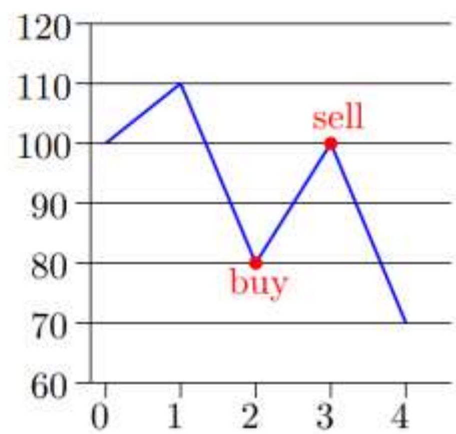
day	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
price	100	113	110	85	105	102	86	63	81	101	94	106	101	79	94	90	97
change		13	-3	-25	20	-3	-16	-23	18	20	-7	12	-5	-22	15	-4	7

Problem

When were the best times to buy and sell?

- A Brute-force Solution
  - Try every possible pair of buy and sell dates (buy precedes sell date)
  - Period of n days has $\binom{n}{2}$ pairs of dates $\binom{n}{2}$ is $\Theta(n^2)$
  - Best we could hope is to evaluate each pair in constant time and approach would take $\Omega(n^2)$
  - Can we do better?

- Difficulty: We don't necessarily buy at the lowest price or sell at the highest price.
- Example:



Problem Transformation

- Consider the array $A[1 \ldots 16]$ of change in price.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
13	-3	-25	20	-3	-16	-23	18	20	-7	12	-5	-22	15	-4	7

Maximum subarray $A[8 \ldots 11]$

- The *maximum subarray* is the contiguous subarray whose elements have the largest sum. Here, it is $A[8 \ldots 11]$.
- So the best times to buy and sell are days 7 and 11.
- We reduced our original problem to:

Problem

Given an array $A[1 \ldots n]$ of n numbers, find the indices p, q such that $1 \leqslant p \leqslant q \leqslant n$ and $\sum_{i=p}^{q} A[i]$ is maximum.

The MAXIMUM SUBARRAY Problem

Brute-force approach to MAXIMUM SUBARRAY

```
procedure MAXIMUM-SUBARRAY(A, 1, n)
    max ← −∞
    for i ← 1, n do
        for j ← i, n do
            sum ← 0
            for k ← i, j do
                sum ← sum + A[k]
            if sum > max then
                max ← sum, p ← i, q ← j
    return p, q, max
```

- Running time? $\Theta(n^3)$.

Applications

The maximum subarray problem has several applications. Some of the well-known applications are in genomic sequence analysis and computer vision. They are used in genomic sequence analysis to identify important segments of protein sequences like GC-rich regions, and regions of high charge. In computer vision, they find their use in detecting the brightest area in bitmap images.