صحت روابط

- صحت روابط زیر را اثبات کنید.
- **توجه:** توابع پیچیدگی را صعودی در نظر بگیرید.

$$(n+\sqrt{n})^3 \in \theta(n^3)$$

$$nlogn \in O(n^n)$$

$$\sum_{i=1}^{n}\sum_{j=i}^{n} j \in \theta(n^3)$$

$$f(n) + o(f(n)) \in \theta(f(n))$$

$$f(n) + g(n) \in O(max(f(n), g(n)))$$

$$f(n) \in O(f^2(n))$$

درستی یا نادرستی

- با فرض مثبت بودن توابع f و g، درستی یا نادرستی روابط زیر را مشخص کنید.
- برای درستی بیان اثبات و برای نادرستی تنها ارائه‌ی یک مثال نقض کافی است.
- **راهنمایی:** برای در نظر گرفتن توابع، همه‌ی توابع را در نظر بگیرید؛ نه صرفاً توابع صعودی.

$$f(n) \in O(g(n)) \rightarrow 2^{f(n)} \in O(2^{g(n)})$$

$$f(n) \in O(f(n)^2)$$

$$f(n) + g(n) \in \theta(min(f(n), g(n)))$$

$$f(n) \in \theta(f(\frac{n}{2}))$$

$$f(n) \in O(g(n)) \rightarrow log(f(n)) \in O(log(g(n)))$$

$$\text{where } \ log(g(n)) \geq 0, f(n) > 0 \ \text{ and } \ f(n) \geq 1 \ \text{ for all sufficiently large n}$$

مقایسه

- الگوریتم‌های A و B دقیقاً $T_A(n)$ و $T_B(n)$ میکرو ثانیه برای حل مسئله‌ای با اندازه‌ی n طول می‌کشند.

$$T_A(n) = 500n^2$$

$$T_B(n) = 0.5n^3$$

- از بین این دو الگوریتم، الگوریتمی را انتخاب کنید که از نظر θ بهتر باشد.
- کوچکترین n_0 ممکن را پیدا کنید که به ازای هر $n \geq n_0$ یک الگوریتم از نظر زمان اجرا بهتر از دیگری باشد.
- اگر $n \leq 10$ باشد، کدام الگوریتم به‌صرفه‌تر است؟

پیچیدگی کدها

- پیچیدگی زمانی تکه کدهای زیر را بدست آورید.

```python
i = n
while i >= 1:
    j = i
    while j <= n:
        print(i)
        j *= 2
    i //= 2
```

```python
i = 2
while i <= n:
    print("*")
    i = i * i
```

- کد زیر را در نظر بگیرید:

```cpp
bool anymatch(int n, int A[][]){
    for(int i=1;i<=n;i++)
        for(int j=1;j<=n;j++)
            for(int k=1;k<=n;k++)
                for(int m=1;m<=n;m++)
                    if(A[i][j]==A[k][m]&&!(i==k&&j==m))
                        return true;
     return false;
}
```

- بهترین و بدترین پیچیدگی زمانی این قطعه کد چیست؟
- در چه صورتی خروجی این قطعه کد $false$ و در چه صورت $true$ خواهد بود؟
- آیا می‌توانید الگوریتم بهینه‌تری ارائه دهید که عملکرد یکسانی با این الگوریتم داشته باشد؟

بازگشت به بازگشتی !

هریک از مسائل زیر با روشی بازگشتی قابل حل هستند. $T(n)$ که زمان اجرای هر یک از این الگوریتم‌هاست را پیدا کرده و پیچیدگی زمانی آن را بدست آورید.(راه حل حتما نوشته شود.)

مسأله اول: برج هانوی

مسئله‌ی برج هانوی $(TowerofHanoi)$ یکی از مسائل تاریخی مشهور است که در مباحث طراحی الگوریتم نیز به آن پرداخته می‌شود.

سه میله‌ی - میله‌ی مبدأ (A) ، میله‌ی کمکی (B) و میله‌ی مقصد (C) - و تعدادی دیسک در میله‌ی مبدأ داریم که به ترتیب از پایین به بالا بزرگ‌ترین دیسک تا کوچک‌ترین دیسک قرار دارند. هدف انتقال تمام دیسک‌ها از این میله به میله‌ی مقصد با رعایت دو شرط زیر است:

- در هر زمان فقط یک دیسک را می‌توان جابجا نمود.
- نباید در هیچ زمانی دیسکی بر روی دیسک با اندازه‌ی کوچکتر قرار بگیرد.

به طور حتم می‌توان با روش آزمون و خطا به نتیجه‌ی مطلوب رسید. اما هدف ما ارائه‌ی الگوریتمی برای انتقال دیسک‌ها با کمترین جابجایی ممکن است. می‌توانید در این لینک بازی را انجام دهید!

مسأله دوم: الگوریتم مِرج سورت

این الگوریتم سورت به این صورت عمل می‌کند که در هر گام آرایه را به دو قسمت مساوی تقسیم کرده، الگوریتم را به ازای هر یک از این دو آرایه‌ی جدید فراخوانی کرده، و سپس آنها را با یکدیگر $merge$ می‌کند. هزینه‌ی $merge$ کردن در هر گام برابر با تعداد اعضای آرایه‌ی مورد نظر در هر گام است. (در گام اول برابر با طول آرایه اصلی، در گام دوم برابر با مجموع طول دو آرایه هریک به طول نصف آرایه اصلی و ...)

مسأله سوم: خیلی گردو...خیلی شکستم...

بازی جدیدی به نام خیلی گردو، خیلی شکستم داریم که به شکل زیر انجام می‌شود.

در هر مرحله یک بازیکن (به عنوان مثال بازیکن "گردو")، باید به اندازه‌ی ۳ برابر شماره‌ی مرحله، بگوید گردو و قدم بردارد. (منطقا بازیکن "شکستم" هم به همین صورت.) هدف شما این است که تعداد کل گام‌های برداشته شده توسط یک بازیکن تا پایان مرحله nام را با یک رابطه بازگشتی بدست آورده و پیچیدگی زمانی این تابع را محاسبه کنید.

واحد اعصاب خورد کن!

- محدودیت زمان: ۱ ثانیه
- محدودیت حافظه: ۵۰ مگابایت

- یک ساختمان غول‌آسا (!) مفروض است با 10000 واحد آپارتمان، و هر کدام از واحدها شناسه‌ای بین 1 تا 10000 دارند.
- تعریف می‌کنیم واحدی حوصله سربر است که همه‌ی رقم‌های آن یکی باشند! به طور مثال واحدهای 55، 2، 9999 و 111 حوصله سربر هستند.
- آقای صورتی وظیفه دارد به تمامی واحدهای حوصله سربر تلفن بزند تا زمانی که یکی از آن‌ها جواب دهد! و این کار را به ترتیب زیر انجام می‌دهد:
  - ابتدا به تمام واحدهای حوصله سربر متشکل از رقم 1، به صورت صعودی زنگ می‌زند: اول 1، سپس 11، سپس 111 و در نهایت 1111.
  - سپس به تمام واحدهای حوصله سربر متشکل از رقم 2، به صورت صعودی زنگ می‌زند: اول 2، سپس 22، سپس 222 و در نهایت 2222.
  - و به همین ترتیب ادامه می‌دهد.
  - در این بین اگر واحد x جواب تلفن را بدهد آقای صورتی بس می‌کند و دیگر به واحدهای بعدی زنگ نمی‌زند!
- می‌خواهیم حساب کنیم که آقای صورتی جمعا چند رقم را فشرده است و از شما می‌خواهیم این مقدار را حساب کنید. به طور مثال اگر واحد 222 جواب بدهد آنگاه خواهیم داشت:

$$1 + 2 + 3 + 4 + 1 + 2 + 3 = 16$$

(چرا که به ترتیب از چپ به راست با واحد های روبرو تماس گرفته است: $(1, 11, 111, 1111, 2, 22, 222)$

ورودی

- خط اول ورودی شامل t (تعداد تست کیس‌ها) است.

$$1 \leq t \leq 36$$

- در خطوط بعدی در هر خط عدد x آمده است که نشان دهنده‌ی شماره آپارتمانی است که پاسخ داده است. همچنین تضمین می‌شود x ارقام یکسانی دارد.

$$1 \leq x \leq 9999$$

خروجی

- به ازای هر تست کیس، تعداد ارقامی که آقای صورتی فشرده است را در یک خط چاپ کنید.

ورودی نمونه

```
4
22
9999
1
777
```

خروجی نمونه

```
13
90
1
66
```

مجموع بازه

- محدودیت زمان: ۵ ثانیه
- محدودیت حافظه: ۵۰ مگابایت

- یک آرایه با طول n را در نظر بگیرید، فرض کنید Q درخواست شامل زوج‌های (i, j) داریم.
- برنامه‌ای بنویسید که به ازای هر درخواست جمع اعداد آرایه بین اندیس‌های i و j را محاسبه کند (اندیس‌ها از صفر شروع می‌شوند).

ورودی

- در خط اول ورودی، عدد طبیعی n آمده است.

$$1 \leq n \leq 50000$$

- در خط دوم ورودی، n عدد حسابی که مقادیر آرایه هستند، آمده است.

$$0 \leq a[i] \leq 100000$$

- در خط سوم ورودی، عدد طبیعی Q که تعداد درخواست هاست، آمده است.

$$1 \leq Q \leq 500000$$

- در Q خط بعدی در هر خط، دو عدد حسابی i و j آمده است.

$$0 \leq i, j < n$$

خروجی

- در خروجی Q خط چاپ کنید که در هر خط جمع مقادیر آرایه بین اندیس‌های i و j آمده باشد.

ورودی نمونه

```
5
1 3 2 7 1
3
4 2
1 1
1 3
```

خروجی نمونه

```
10
3
12
```

سیم های پیچیده

- محدودیت زمان: ۱ ثانیه
- محدودیت حافظه: ۱۲۸ مگابایت
- پاسخ گویی به یکی از دو سوال Memories of Dust و سیم های پیچیده اجباریست و یکی از آن ها به عنوان تمرین امتیازیست.

تعداد n خانه در نقاط صحیح مختصات قرار دارند. می‌خواهیم بین هر دو خانه یک سیم تلفن بکشیم. میزان سیم مورد نیاز برای وصل کردن دو خانه به اندازه‌ی فاصله‌ی منهتن آن‌ها است. برنامه‌ای بنویسید که طول بزرگترین سیم موردنیاز را پیدا کند. فاصله‌ی منهتن بین دو نقطه‌ی (x_i, y_i) و (x_j, y_j) برابر است با:

$$|x_i - x_j| + |y_i - y_j|$$

ورودی

در خط اول ورودی ابتدا عدد n داده می‌شود. سپس در n خط بعد در هر خط دو عدد که نشان‌دهنده‌ي مختصات خانه‌ی iام است داده می‌شود.

$$1 \leq n \leq 550\ 000$$

$$-1\ 000\ 000 \leq x_i, y_i \leq 1\ 000\ 000$$

خروجی

طول بزرگترین سیم مورد استفاده را چاپ کنید. (منظور از بزرگترین سیم، ماکسیمم فاصله بین جفت نقاطی است که در هر خط داده شده)

مثال

ورودی نمونه

```
5
2 2
4 6
3 8
9 2
5 5
```

خروجی نمونه

```
12
```

** توجه: ** برنامه شما باید از مرتبه n باشد و در صورت حل عادی با مرتبه n^2 ، دچار تایم لیمیت خواهید شد!

Memories of Dust

- محدودیت زمان: ۱ ثانیه
- محدودیت حافظه: ۲۵۶ مگابایت
- پاسخ گویی به یکی از دو سوال Memories of Dust و سیم های پیچیده اجباریست و یکی از آن ها به عنوان تمرین امتیازیست.

توضیح تصویر

نواین اِس($9S$)، به تازگی متوجه شده که مهارت‌هایش در هک کردن ماشین‌ها در حال ضعیف شدن هستند و برای همین تصمیم گرفته است که برای تمرین، تعدادی ماشین را به طور متوالی هک کند. او n ماشین در صف دارد که باید برای هک کردن آن ها تلاش کند، اما مشکلی که وجود دارد این است که همه ماشین ها به قوی‌تر شدن اون کمک نمی‌کنند و هک کردن برخی می‌تواند از قدرت او بکاهد. به طور کلی هک کردن ماشین iام قدرت a_i را به نواین اِس اضافه میکند که این قدرت میتواند یک عدد صحیح منفی یا نامنفی باشد،مشکل دیگری که نواین اِس دارد این است که اگر هک کردن را شروع کند، دوست ندارد بیش از یک بار متوقف شود، در نتیجه میخواهد فقط یک بازه متوالی از ماشین ها را هک کند. حال به توجه به شرایط تمرین، حداکثر قدرتی که نواین اِس می‌تواند از این تمرین بدست آورد، چقدر است؟

ورودی

در خط اول ورودی عدد n آمده است که تعداد ماشین‌ها را نشان می‌دهد.
در خط بعدی، n عدد آمده که عدد i ام نشان‌دهنده قدرتی است که هک کردن ماشین i ام به نواینِ اس اضافه می‌کند. دقت کنید که ماشین‌ها قابل جابجایی نیستند.

$$1 \leq n \leq 10^6$$

$$-10^7 \leq a_i \leq 10^7$$

خروجی

خروجی برنامه شما باید شامل یک عدد که که نشان‌دهنده حداکثر قدرتی است که نایِن اِس می‌تواند با هک کردن بازه‌ای متوالی از ماشین‌ها بدست آورد، باشد. توجه کنید که اگر بازه‌ای وجود نداشته باشد که قدرت نایِن اِس را افزایش دهد، او هیچ ماشینی را هک نمی‌کند و خروجی صفر خواهد بود.

مثال

ورودی نمونه ۱

```
7
-3 1 3 4 -2 0 1
```

خروجی نمونه ۱

```
8
```

ماکسیمم قدرتی که نایِن اِس می‌تواند بدست بیاورد، از هک کردن بازه شامل ماشین‌های دوم، سوم و چهارم است که در مجموع به او قدرت ۸ را اضافه می‌کنند.

ورودی نمونه ۲

```
5
4 4 4 -2 10
```

خروجی نمونه ۲

```
20
```

در این نمونه،بیشترین قدرت از بازه شامل همه ۵ عدد بدست می آید.