محدوده حداقل پرس و جو (تجزیه ریشه مربعی و جدول پراکنده)

فهرست مطالب

محدوده حداقل پرس و جو (تجزیه ریشه مربعی و جدول پراکنده) 1
مقدمه 1
روش 1 (راه حل ساده) 1
روش 2 (تجزیه ریشه مربع) (Square Root Decomposition) 2
روش 3 (الگوریتم جدول پراکنده) Sparse Table Algorithm 3
نتیجه 4
بیوانفورماتیک تجزیه و تحلیل توالی 5
راه اندازی توسعه Biopython 5
خواندن دنباله در قالب FASTA 6

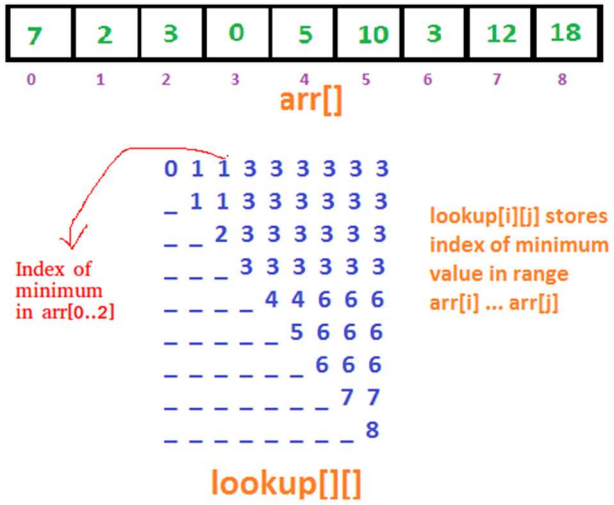
مقدمه

ما یک آرایه آرایه داریم [0. . . n-1] ما باید بتوانیم کمترین مقدار را از شاخص L (شروع پرس و جو) تا R (پایان پرس و جو) که 0 <= L <= R <= n-1 به طور کارآمد پیدا کنیم. شرایطی را در نظر بگیرید که پرس و جوهای دامنه زیادی وجود دارد. یک راه حل ساده اجرای حلقه از L به R و یافتن حداقل عنصر در محدوده داده شده است. این راه حل در بدترین حالت به زمان O (n) نیاز دارد. روش دیگر استفاده از درخت Segment است. با درخت بخش ، زمان پیش پردازش O (n) و زمان برای حداقل پرس و جو برای دامنه O (Logn) است. فضای اضافی مورد نیاز O (n) برای ذخیره درخت قطعه است. Segment tree همچنین در زمان O (ورود به سیستم) به روزرسانی می کند.

وقتی هیچ عملیاتی برای بروزرسانی وجود ندارد و تعداد محدودی پرس و جو وجود دارد ، چگونه زمان پرس و جو را بهینه کنیم؟ در زیر روش های مختلف وجود دارد.

روش 1 (راه حل ساده)

یک راه حل ساده ایجاد جستجوی آرایه 2D است [] [] جایی که جستجوی ورودی [j] [i] حداقل مقدار را در arr ar [i..j] ذخیره می کند. حداقل دامنه داده شده اکنون می تواند در زمان O (1) محاسبه شود.



این روش از کوئری ها در O (1) پشتیبانی می کند ، اما پیش پردازش O (n2) زمان می برد. همچنین ، این روش به فضای اضافی O (n2) نیاز دارد که ممکن است برای آرایه های ورودی بزرگ بسیار بزرگ شود.

روش 2 (تجزیه ریشه مربع) (Square Root Decomposition)

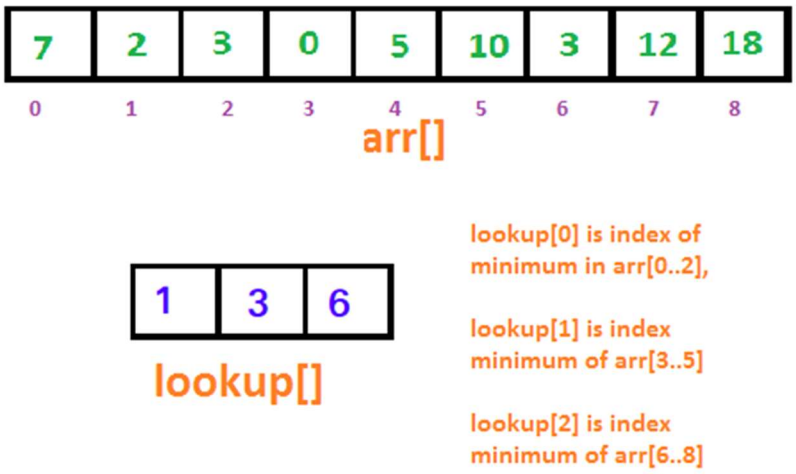
ما می توانیم از Square Root Decompositions برای کاهش فضای مورد نیاز در روش فوق استفاده کنیم.

پیش پردازش:

1) محدوده [0 ، n-1] را به بلوک های مختلف √n تقسیم کنید.

2) حداقل هر بلوک در اندازه را محاسبه کنید و نتایج را ذخیره کنید.

پیش پردازش زمان O (n) = O (n * √n*) و فضای O ()n را می گیرد.



پرس و جو- Query:

1) برای پرس و جو از یک محدوده [L، R] ، ما حداقل تمام بلوک های موجود در این محدوده را می گیریم. برای بلوک های گوشه چپ و راست که ممکن است تا حدی با دامنه داده شده همپوشانی داشته باشند ، ما آنها را به صورت خطی اسکن می کنیم تا حداقل را پیدا کنیم.

پیچیدگی زمانی پرس و جو O (n) است. توجه داشته باشید که حداقل بلوک میانی به طور مستقیم قابل دسترسی است و حداکثر بلوک های میانی O (√n) وجود دارد. حداکثر دو بلوک گوشه ای وجود دارد که ممکن است مجبور به اسکن آنها باشیم ، بنابراین ممکن است مجبور شویم عناصر بلوک گوشه ای O (n * 2) را اسکن کنیم. بنابراین ، پیچیدگی کلی زمان O (n) است.

به تکنیک تجزیه Sqrt (یا ریشه مربع) مراجعه کنید | 1 (مقدمه) را برای جزئیات تنظیم کنید.

روش 3 (الگوریتم جدول پراکنده) Sparse Table Algorithm

راه حل فوق فقط به فضای O (n) نیاز دارد اما برای پرس و جو به O (n) زمان نیاز دارد. روش جدول پراکنده از زمان پرسش O (1) با فضای اضافی O (n Log n) پشتیبانی می کند.

ایده این است که حداقل همه زیرشاخه های اندازه 2j را که j از 0 تا Log n متغیر است ، از پیش محاسبه کنیم. مانند روش 1 ، ما یک جدول جستجو ایجاد می کنیم. در اینجا جستجوی [j] [i] شامل حداقل دامنه شروع از i و اندازه 2j است. به عنوان مثال جستجوی [0] [3] شامل حداقل دامنه [0 ، 7] (شروع با 0 و اندازه 23)

پیش پردازش:

چگونه این جدول جستجو را پر کنیم؟ ایده ساده است ، با استفاده از مقادیر قبلاً محاسبه شده ، روشی از پایین به بالا را پر کنید.

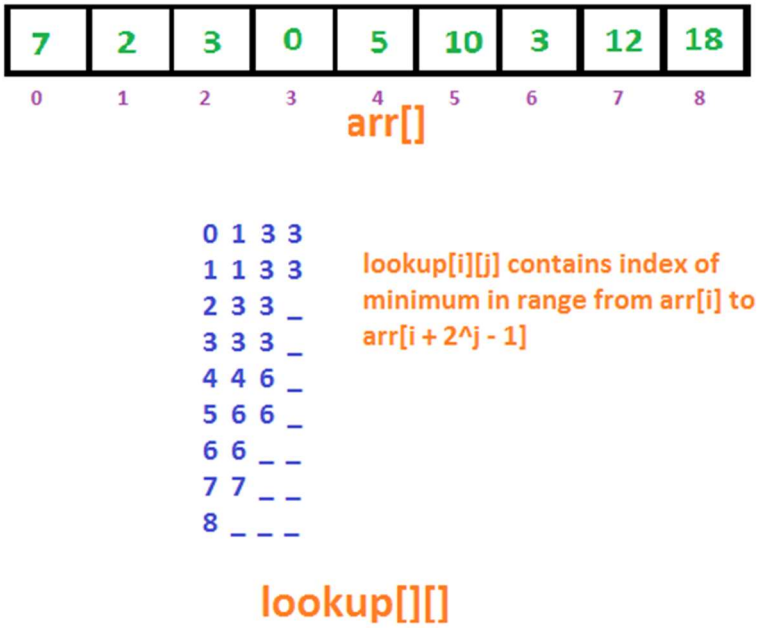
به عنوان مثال ، برای یافتن حداقل دامنه [0 ، 7] ، می توانیم از حداقل دو زیر استفاده کنیم. بر اساس مثال فوق ، فرمول زیر است:

الف) حداقل دامنه [0 ، 3]

ب) حداقل دامنه [4 ، 7]

```
// If arr[lookup[0][2]] <=  arr[lookup[4][2]],
// then lookup[0][3] = lookup[0][2]
If arr[lookup[i][j-1]] <= arr[lookup[i+2^(j-1)][j-1]]
   lookup[i][j] = lookup[i][j-1]

// If arr[lookup[0][2]] >  arr[lookup[4][2]],
// then lookup[0][3] = lookup[4][2]
Else
   lookup[i][j] = lookup[i+2^(j-1)][j-1]
```



پرس و جو:

برای هر محدوده دلخواه [l، R] ، ما باید از محدوده هایی استفاده کنیم که توان آنها 2 باشد. هدف این است که از نزدیکترین قدرت 2 استفاده کنیم. ما همیشه باید حداکثر یک مقایسه انجام دهیم (حداقل دو دامنه را مقایسه کنید که قدرت 2 هستند) یک دامنه با L شروع می شود و با "L + نزدیکترین نیرو از 2" پایان می یابد. دامنه دیگر به R ختم می شود و با "R - همان نزدیکترین نیرو از 2 + 1" شروع می شود. به عنوان مثال ، اگر دامنه داده شده (2 ، 10) باشد ، حداقل دو دامنه (2 ، 9) و (3 ، 10) را مقایسه می کنیم.

بر اساس مثال فوق ، فرمول زیر است:

```
// For (2,10), j = floor(Log2(10-2+1)) = 3
j = floor(Log(R-L+1))

// If arr[lookup[0][3]] <=  arr[lookup[3][3]],
// then RMQ(2,10) = lookup[0][3]
If arr[lookup[L][j]] <= arr[lookup[R-(int)pow(2,j)+1][j]]
   RMQ(L, R) = lookup[L][j]

// If arr[lookup[0][3]] >  arr[lookup[3][3]],
// then RMQ(2,10) = lookup[3][3]
Else
   RMQ(L, R) = lookup[R-(int)pow(2,j)+1][j]
```

نتیجه

از آنجا که ما فقط یک مقایسه انجام می دهیم ، پیچیدگی زمانی پرس و جو O (1) است. بنابراین روش جدول پراکنده از عملیات پرس و جو در زمان O (1) با زمان پیش پردازش O (n Log n) و فضای O (n Log n) پشتیبانی می کند.

بیوانفورماتیک تجزیه و تحلیل توالی

یوانفورماتیک رشته ای است که ترکیبی از دو زمینه اصلی است: داده های بیولوژیکی (توالی ها و ساختارهای پروتئین ها ، DNA، RNA ها و سایر موارد) و انفورماتیک (علوم کامپیوتر ، آمار ، ریاضیات و مهندسی). به همین دلیل به آن رشته میان رشته ای می گویند.
استفاده معمول از بیوانفورماتیک شامل شناسایی ژن ها و SNP ها است.

اهداف بیوانفورماتیک عبارتند از:

هم ترازی ترتیب.

یافتن ژن

کشف دارو و طراحی دارو

ترازبندی و پیش بینی ساختار پروتئین. و غیره.

همانطور که متوجه شدید ، بیوانفورماتیک یک رشته بزرگ است که شامل مناطق مختلف و عملیات مربوطه است. در این پست مقدماتی ، ما به طور خلاصه در مورد تجزیه و تحلیل توالی بحث می کنیم.

در بیوانفورماتیک ، تجزیه و تحلیل توالی فرآیند قرار دادن توالی DNA، RNA یا پپتید در هر یک از طیف وسیعی از روش های تحلیلی برای درک ویژگی ها ، عملکرد ، ساختار یا تکامل آن است. تعیین توالی فرآیند یافتن ساختار اولیه است خواه DNA، RNA باشد. توالی DNA در واقع تعیین توالی اسید نوکلئیک است. این کمک می کند تا ترتیب چهار پایه آدنین (A) ، گوانین (G) ، سیتوزین (C) و تیمین (T) در یک رشته DNA را دریابید.

تعیین توالی با کمک ماشین های توالی انجام می شود. پس از توالی ، داده ها برای پردازش بیشتر در دسترس هستند. یک مثال متنی از توالی DNA مانند زیر است:

```
CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGATGAGACCGTGGAATAAACGATCGAGTG
AATCCGGAGGACCGGTGTACTCAGCTCACCGGGGGCATTGCTCCCGTGGTGACCCTGATTTGTTGTTGGG
CCGCCTCGGGAGCGTCCATGGCGGGTTTGAACCTCTAGCCCGGCGCAGTTTGGGCGCCAAGCCATATGAA
AGCATCACCGGCGAATGGCATTGTCTTCCCCAAAACCCGGAGCGGCGGCGTGCTGTCGCGTGCCCAATGA
```

تعیین توالی DNA روش تعیین ترتیب نوکلئوتید در DNA است.

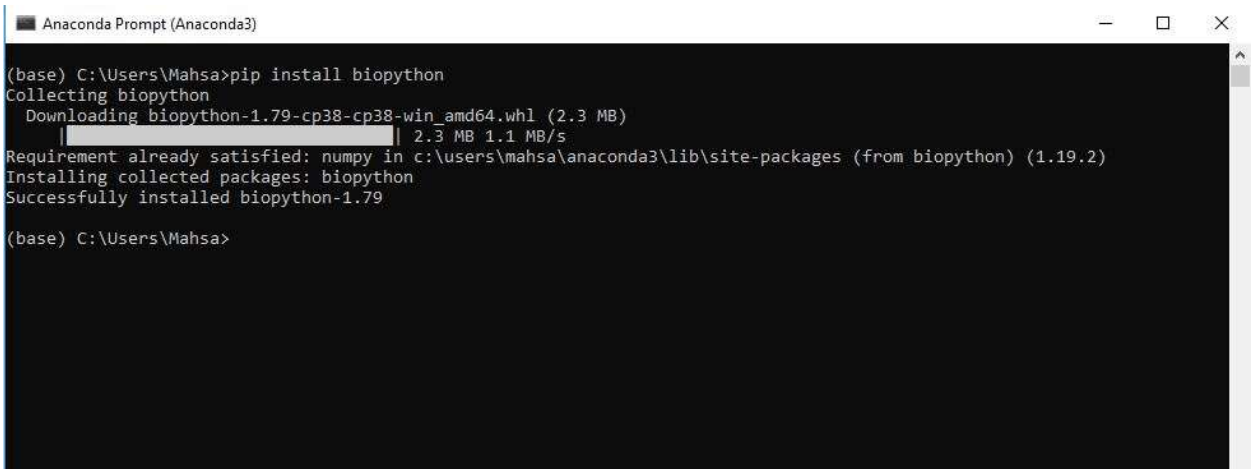
تعیین توالی RNA روشی برای یافتن مقدار RNA در یک نمونه بیولوژیکی است.

تعیین توالی پروتئین روشی برای تعیین توالی اسید آمینه کل یا بخشی از پروتئین یا پپتید است.

راه اندازی توسعه Biopython

ما قصد داریم از Biopython ، کتابخانه ای برای مقابله با محاسبات بیولوژیکی و تجزیه و تحلیل توالی ها استفاده کنیم. در صورت استفاده از توزیع مبتنی بر Conda ، می توانید از pip یا conda استفاده کنید.

```
pip install biopython
```



خواندن دنباله در قالب FASTA

خوب ، بنابراین ما می خواهیم یک توالی DNA را بخوانیم که در قالب FASTA موجود است. FASTA یک قالب مبتنی بر متن برای نشان دادن توالی های مختلف است که در کدهای تک حرف نشان داده می شوند. یک قالب معمول FASTA مانند زیر است:

```
;LCBO - Prolactin precursor - Bovine
; a sample sequence in FASTA format
>gi|2765658|emb|Z78533.1|CIZ78533 C.irapeanum 5.8S rRNA gene and ITS1 and ITS2 DNA
CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGATGAGACCGTGGAATAAACGATCGAGTG
AATCCGGAGGACCGGTGTACTCAGCTCACCGGGGGCATTGCTCCCGTGGTGACCCTGATTTGTTGTTGGG
CCGCCTCGGGAGCGTCCATGGCGGGTTTGAACCTCTAGCCCGGCGCAGTTTGGGCGCCAAGCCATATGAA
AGCATCACCGGCGAATGGCATTGTCTTCCCCAAAACCCGGAGCGGCGGCGTGCTGTCGCGTGCCCAATGA
ATTTTGATGACTCTCGCAAACGGGAATCTTGGCTCTTTGCATCGGATGGAAGGACGCAGCGAAATGCGAT
AAGTGGTGTGAATTGCAAGATCCCGTGAACCATCGAGTCTTTTGAACGCAAGTTGCGCCCGAGGCCATCA
GGCTAAGGGCACGCCTGCTTGGGCGTCGCGCTTCGTCTCTCTCCTGCCAATGCTTGCCCGGCATACAGCC
AGGCCGGCGTGGTGCGGATGTGAAAGATTGGCCCCTTGTGCCTAGGTGCGGCGGGTCCAAGAGCTGGTGT
TTTGATGGCCCGGAACCCGGCAAGAGGTGGACGGATGCTGGCAGCAGCTGCCGTGCGAATCCCCCATGTT
GTCGTGCTTGTCGGACAGGCAGGAGAACCCTTCCGAACCCCAATGGAGGGCGGTTGACCGCCATTCGGAT
GTGACCCCAGGTCAGGCGGGGGCACCCGCTGAGTTTACGC

>MCHU - Calmodulin - Human, rabbit, bovine, rat, and chicken
ADQLTEEQIAEFKEAFSLFDKDGDGTITTKELGTVMRSLGQNPTEAELQDMINEVDADGNGTID
FPEFLTMMARKMKDTDSEEEIREAFRVFDKDGNGYISAAELRHVMTNLGEKLTDEEVDEMIREA
DIDGDGQVNYEEFVQMMTAK*

>gi|5524211|gb|AAD44166.1| cytochrome b [Elephas maximus maximus]
LCLYTHIGRNIYYGSYLYSETWNTGIMLLLITMATAFMGYVLPWGQMSFWGATVITNLFSAIPYIGTNLV
EWIWGGFSVDKATLNRFFAFHFILPFTMVALAGVHLTFLHETGSNNPLGLTSDSDKIPFHPYYTIKDFLG
LLILILLLLLALLSPDMLGDPDNHMPADPLNTPLHIKPEWYFLFAYAILRSVPNKLGGVLALFLSIVIL
GLMPFLHTSKHRSMMLRPLSQALFWTLTMDLLTLTWIGSQPVEYPYTIIGQMASILYFSIILAFLPIAGX
IENY
```

نظرات بعد از نقطه ویرگول اضافه می شوند. هر دنباله با یک نماد> جدا می شود. مثالی که در بالا ذکر شد دارای 3 توالی است که در یک پرونده FASTA ارائه شده است. این فقط یک نمونه است در غیر این صورت لازم نیست.