TOKI Regular Open Contest #16 Editorial

rama_pang Pyqe Berted dascogabriel

22 November 2020

(English version is available on page 8.)

	Judul	Pembuat Soal	Penyiapan	Editorialis
A	Teka-teki Terbingung	Pyqe	Pyqe	rama_pang
B	Pelajaran Pangan	Berted	Berted	Berted
C	Kalkulasi Keimutan	Pyqe	Pyqe	rama_pang
D	Keramahan Keramaian	Berted	Berted	rama_pang
E	Kali-Kali Kelipatan	rama_pang	prabowo	rama_pang
F	Radius Radio	dascogabriel	dascogabriel	dascogabriel
G	Bandar Bundar	rama_pang	rama_pang	rama_pang

A. Teka-teki Terbingung

Perhatikan bahwa segi empat dengan panjang sisi 1 dan N memiliki luas sebesar $1 \cdot N = N$. Maka kita cukup mengeluarkan panjang sisi berupa 1 dan N.

Kompleksitas waktu: $O(1)$

B. Pelajaran Pangan

Untuk selanjutnya, anggap barisan A telah diurutkan dan penomoran dimulai dari 0 untuk mempermudah pembahasan.

Jika Djodjo ingin berpindah dari x ke y searah jarum jam, maka waktu minimum yang diperlukan Djodjo adalah $y - x$ jika $y > x$ atau $M - y + x$ jika $y < x$.

Observasi. *Dalam cara optimal, Djodjo mulai bergerak dari A_i sampai $A_{(i-1) \bmod N}$ (posisi di sebelah kiri A_i) searah jarum jam untuk suatu indeks i.*

Sehingga, kita cukup menghitung waktu yang diperlukan untuk setiap indeks i dan mengambil nilai yang terkecil.

Kompleksitas waktu: $O(N \log N)$

C. Kalkulasi Keimutan

Observasi. *Array A tidak imut jika tidak terdapat elemen dengan nilai M.*

Observasi. *Array A tidak imut jika terdapat dua elemen dengan nilai M yang bersebelahan.*

Bukti. Jika terdapat dua elemen dengan nilai M yang bersebelahan, maka kedua elemen tersebut tidak bisa dihapus. Sementara, array dikatakan imut jika kita bisa menghapus semua kecuali 1 elemen dengan nilai M. □

Observasi. *Semua array A yang tidak memenuhi dua hal di atas adalah array imut.*

Bukti. Selama banyak elemen dengan nilai M lebih dari 1, hapus elemen tersebut. Karena tidak ada elemen dengan nilai M yang bersebelahan, hal ini pasti bisa dilakukan. Setelah tersisa satu elemen dengan nilai M, kita dapat menghapus elemen-elemen di kiri dan kanan M karena elemen tersebut lebih besar dari semua elemen lain dalam array yang tersisa. □

Kita dapat menggunakan *dynamic programming* untuk mencari banyaknya array yang tidak memiliki elemen dengan nilai M yang bersebalahan. Anggap $dp[n][0]$ adalah banyaknya array dengan panjang n dengan elemen ke-n memiliki nilai bukan M, dan $dp[n][1]$ adalah banyaknya array dengan panjang n dengan elemen ke-n memiliki nilai sama dengan M.

Base case dari dp adalah $dp[0][0] = 1$ dan $dp[0][1] = 0$.

$dp[n][0] = (M-1) \cdot (dp[n-1][0] + dp[n-1][1])$, karena nilai elemen ke-n tidak sama dengan M sehingga kita bisa menyambungkannya ke semua array dengan panjang $n-1$ yang valid.

$dp[n][1] = dp[n-1][0]$, karena jika kita meletakkan nilai M pada elemen ke-n berarti nilai elemen ke-$(n-1)$ tidak boleh sama dengan M.

Sehingga, banyaknya array dengan panjang N yang tidak ada dua elemen dengan nilai M yang bersebelahan adalah $dp[N][0] + dp[N][1]$. Namun, array dikatakan imut apabila terdapat minimal sebuah elemen dengan nilai M. Kita harus mengurangi jawaban dengan banyaknya array yang tidak memiliki nilai M, yaitu $(M-1)^N$.

Maka, banyaknya array yang imut adalah $dp[N][0] + dp[N][1] - (M-1)^N$. dp memiliki $O(N)$ states dan untuk setiap state, transisi dapat dilakukan dalam $O(1)$. Sehingga, dp dapat dihitung dalam $O(N)$.

Kompleksitas waktu: $O(N)$

D. Keramahan Keramaian

Kita akan melakukan *binary search* pada jawaban. Anggap kita ingin mengecek apakah $answer \geq mid$. Tambahkan edge (u,v) jika $A[u][v] < mid$, dan sebut ini sebagai graf G. Perhatikan bahwa jika dua verteks pada G terhubung dengan sebuah edge, maka kedua verteks tersebut harus berada pada kelompok yang berbeda.

Berarti, untuk setiap verteks u kita ingin mewarnainya dalam salah satu dari dua warna sehingga tidak ada sepasang verteks dengan warna yang sama yang terhubung dengan sebuah edge. Hal ini sama saja dengan menentukan apakah G adalah sebuah graf bipartit.

Perhatikan bahwa dalam *binary search*, hanya terdapat $O(N^2)$ nilai-nilai yang menjadi kandidat jawaban, maka *binary search* dapat dilakukan hanya untuk $O(N^2)$ nilai-nilai tersebut.

Kompleksitas waktu: $O(N^2 \log N)$

E. Kali-Kali Kelipatan

Definisi. *$f(i,j)$ adalah kelipatan persekutuan terkecil dari $A[i],A[i+1],..,A[j]$.*

Definisi. *$cnt_p[i]$ adalah bilangan x terbesar sedemikian sehingga $p^x \mid A[i]$.*

Definisi. *$rangemax_p(i,j)$ adalah nilai maksimum dari $cnt_p[i],cnt_p[i+1],..,cnt_p[j]$.*

Teorema. $\prod_{i=1}^{N}\prod_{j=i}^{N} f(i,j) = \prod_{p\in prime} p^{\sum_{i=1}^{N}\sum_{j=i}^{N} rangemax_p(i,j)}$

Kita akan menyelesaikan persoalan untuk setiap bilangan prima p secara terpisah dengan menghitung $\sum_{i=1}^{N}\sum_{j=i}^{N} rangemax_p(i,j)$. Kita dapat menghitung $\sum_{i=1}^{N}\sum_{j=i}^{N} rangemax_p(i,j)$ dalam $O(N)$ atau $O(N\log A_{max})$ dengan menggunakan stack. Iterasi r dari 1 sampai N sambil menyimpan sebuah stack berisi pair (mx,id), dengan $stack[i].mx \geq stack[i+1].mx$ dan id adalah indeks l terbesar sedemikian sehingga $rangemax_p(i,r) \geq mx$ untuk $1 \leq i \leq l$.

Observasi. *Suatu bilangan x memiliki $O(\log x)$ faktor prima.*

Korolari. *$rangemax_p(i,r) \leq O(\log A_{max})$.*

Korolari. *$\sum_{p\in prime}\sum_{i=1}^{N} cnt_p[i] = O(N\log A_{max})$.*

Dengan menggunakan stack ini, kita dapat menghitung $\sum_{i=1}^{r} rangemax_p(i,r)$ dengan mengiterasi setiap elemen dalam stack. Karena banyaknya elemen dalam stack maksimal adalah $O(\log A_{max})$, algoritme berjalan dalam $O(N\log A_{max})$. Untuk menghilangkan faktor $\log A_{max}$, kita dapat menyimpan variabel $power\_sum = \sum_{i=1}^{r} rangemax_p(i,r)$ secara terpisah.

Untuk memperbarui stack dari r ke $r+1$, kita cukup melakukan pop pada elemen teratas stack selama $stack.top().mx \leq cnt_p[r]$, kemudian push pair $(cnt_p[r],r)$. Dalam solusi $O(N)$, kita juga memperbarui $power\_sum$ setiap kali kita melakukan pop atau push ke dalam stack.

Algoritme di atas dapat dioptimisasi dengan tidak memproses indeks i jika $cnt_p[i] = 0$. Dengan demikian, karena $\sum_{p\in prime}\sum_{i=1}^{N} cnt_p[i] = O(N\log A_{max})$ maka kompleksitas akhir dari solusi ini adalah $O(N\log A_{max})$ atau $O(N\log^2 A_{max})$.

Terdapat juga solusi tanpa harus melakukan optimisasi tersebut.

Observasi. *Jika $p > \sqrt{A_{max}}$ maka $0 \leq cnt_p[i] \leq 1$.*

Teorema. *Anggap $\pi(x)$ adalah sebuah fungsi yang menghitung banyaknya prima $p \leq x$, maka $\pi(x) = O(\frac{x}{\log x})$.*

Kita dapat menggunakan algoritme di atas untuk semua prima $p \leq \sqrt{A_{max}}$. Karena terdapat $O(\frac{\sqrt{A_{max}}}{\log A_{max}})$ prima $p \leq \sqrt{A_{max}}$, kompleksitas bagian ini adalah $O(N\frac{\sqrt{A_{max}}}{\log A_{max}})$ atau $O(N\sqrt{A_{max}})$. Untuk $p > \sqrt{A_{max}}$, kita cukup menghitung ada berapa banyak subarray yang memiliki elemen 1 dalam subarray tersebut. Hal ini dapat dilakukan dalam $O(\sum_{i=1}^{N} cnt_p[i])$ dengan mudah.

Kompleksitas waktu: $O((N+A_{max})\log A_{max})$

F. Radius Radio

Definisi. *Sebuah menara radio j dikatakan berada dalam mask jika dan hanya jika bit ke-j dari mask menyala.*

Definisi. *$f(mask)$ adalah banyak rumah yang dapat dijangkau oleh **semua** menara radio dalam mask.*

Definisi. *$g(mask)$ adalah banyak rumah yang dapat dijangkau oleh **minimal salah satu** menara radio dalam mask.*

Definisi. *Sebuah rumah dikatakan terjangkau sempurna oleh himpunan menara radio S jika setiap menara radio dalam S dapat menjangkau rumah tersebut.*

Perhatikan bahwa jika sebuah rumah terjangkau sempurna oleh himpunan menara radio S, maka rumah tersebut juga terjangkau sempurna oleh semua subhimpunan dari S. Untuk setiap rumah i, kita mencari himpunan menara radio S maksimal yang dapat menjangkau rumah tersebut dan menyimpannya dalam array H. Lalu, kita bisa menghitung $f(mask)$ dengan menggunakan dynamic programming sum over superset pada array H.

Setelah itu, kita dapat menghitung nilai $g(mask)$ menggunakan dynamic programming sum over subset dan prinsip inklusi-eksklusi dengan nilai-nilai $f(mask)$.

Kompleksitas waktu: $O(NM + M2^M)$

G. Bandar Bundar

Untuk selanjutnya, kita mengubah edge busur W_i menjadi edge garis lurus $(i, i+1)$ dengan weight W_i. Jika ada edge duplikat, pilihlah yang memiliki weight lebih rendah.

Definisi. *Sebuah poligon ukuran N adalah sebuah poligon yang memiliki N verteks dan N sisi.*

Definisi. *Garis bukan sisi dari sebuah poligon adalah garis lurus yang menghubungkan dua verteks pada poligon yang tidak bersebelahan.*

Bentuk graf awal adalah sebuah poligon ukuran N dengan M edges. Edges meliputi garis sisi dan garis bukan sisi.

Lemma. *Untuk sebuah poligon konveks dengan N sisi dan M garis bukan sisi yang tidak saling berpotongan, terdapat poligon reguler yang isomorfik dengan N sisi dan M garis bukan sisi yang tidak saling berpotongan.*

Lemma. *Kita dapat mempartisi poligon reguler ukuran N menjadi $N-2$ buah segitiga dengan melakukan operasi berikut ini selama masih bisa dilakukan: menghubungkan 2 verteks pada poligon dengan sebuah garis lurus sedemikian sehingga tidak ada garis yang saling berpotongan. Hal ini berlaku untuk semua urutan dan konfigurasi operasi yang mungkin.*

Bukti. Kita akan membuktikan melalui induksi. Anggap $f(N)$ merupakan banyak segitiga untuk poligon reguler ukuran N.

Untuk $N = 3$, maka poligon tersebut merupakan segitiga sehingga $f(3) = 3-2 = 1$ terpenuhi.

Untuk $N > 3$, asumsikan $f(x) = x-2$ berlaku untuk semua $x < N$. Pasti terdapat minimal satu garis bukan sisi yang membagi poligon reguler menjadi dua poligon konveks yang lebih kecil dengan ukuran L dan R. Maka $L+R = N+2$, karena garis pembagi berada di kedua poligon dan $f(N) = f(L)+f(R)$. Karena $3 \leq L,R$, maka $L,R < N$. Sehingga $f(N) = (L-2)+(R-2) = L+R-4$. Subsitusikan $L+R = N+2$ untuk mendapatkan $f(N) = N+2-4 = N-2$. □

Teorema. *$M \leq 2N-3$ dan untuk konfigurasi M edges apa pun, kita bisa menambahkan sebanyak $2N-3-M$ edges dengan weight ∞, sehingga pada akhirnya terdapat $2N-3$ edges pada poligon reguler ukuran N. Hal ini tidak akan mempengaruhi jawaban.*

Bukti. Setiap edge baru akan memiliki weight ∞, sehingga tidak akan digunakan dalam jalur optimal. Sekarang, anggap terdapat poligon reguler ukuran N. Tiap verteks dinomori 1 sampai N searah jarum jam. Untuk setiap edge (u,v), gambar sebuah garis lurus dari verteks u ke verteks v.

Perhatikan bahwa kondisi poligon sekarang adalah pertengahan dari proses mempartisikan poligon reguler ukuran N menjadi $N-2$ segitiga yang belum selesai. Karena pada akhirnya akan terdapat sebanyak $N-2$ segitiga, berarti poligon reguler akan dibagi dengan $N-3$ garis bukan sisi. Sehingga, terdapat N garis sisi dan $N-3$ garis bukan sisi. Dari $2N-3$ garis total, kita telah meletakkan M garis. Untuk melengkapinya kita perlu menambahkan $2N-3-M$ garis baru. □

Definisi. *$path(u,v)$ adalah jalur optimal dari verteks u ke v.*

Definisi. *$d(u,v)$ adalah panjang jalur optimal dari verteks u ke v.*

Sekarang kita dapat berasumsi bahwa $M = 2N-3$ dengan menambahkan edges tambahan dengan weight ∞. Untuk mendapatkan edges tambahan tersebut, pertama inisialisasi sebuah himpunan $S = \{1,2,..,N\}$. Cari sebuah verteks $u \in S$ dengan *degree* 2 dan sambungkan kedua verteks yang bersebelahan dengan u dengan sebuah garis, lalu hapus u dari S. Ulangi operasi tersebut selama $|S| > 3$.

Selanjutnya kita akan menghitung $\sum_{u=1}^{N-1}\sum_{v=u+1}^{N} d(u,v)$ dengan *divide and conquer*. *Base case* adalah ketika $N = 3$ dan dapat ditangani dengan mudah. Untuk selanjutnya, asumsi $N > 3$.

Dalam sebuah poligon reguler dengan edges, kita dapat memilih salah satu garis bukan sisi dan memisahkan poligon menjadi dua poligon yang lebih kecil. Anggap edge yang menjadi pembagi adalah (X,Y). L dan R adalah himpunan dari verteks-verteks yang berada pada bagian yang berbeda setelah poligon dibagi oleh (X,Y).

Terdapat beberapa permasalahan sebelum kita dapat melakukan rekursi. Bisa saja $path(u,v)$ dengan $u,v \in L$ melalui suatu verteks $w \in R$. Namun jika demikian, pasti jalur tersebut melewati verteks X dan Y. Sehingga, sebelum kita melakukan rekursi kita dapat menjalankan algoritme Dijkstra untuk mencari jalur optimal dari X menuju Y dan menggantikan weight (X,Y) dengan panjang jalur optimal tersebut. Dengan demikian, setiap bagian menjadi independen dari satu sama lain.

Sekarang, kita perlu menghitung jumlah dari $d(u,v)$ untuk $u \in L$ dan $v \in R$. $path(u,v)$ pasti melewati salah satu dari X ataupun Y. Perhatikan bahwa $d(u,v) = \min(d(X,u)+d(X,v), d(Y,u)+d(Y,v))$. Kita akan menghitung jumlah dari pasangan $d(u,v)$ ketika $d(X,u)+d(X,v) \leq d(Y,u)+d(Y,v)$.

$d(X,u)+d(X,v) \leq d(Y,u)+d(Y,v)$ dapat ditulis kembali menjadi $d(X,u)-d(Y,u) \leq d(Y,v)-d(X,v)$. Lakukan sortir verteks-verteks pada R berdasarkan $d(Y,v)-d(X,v)$. Setelah itu, untuk setiap verteks $u \in L$, kita dapat mencari jumlah dari $d(X,v)$ untuk semua $v \in R$ yang memenuhi $d(X,u)-d(Y,u) \leq d(Y,v)-d(X,v)$ dengan mudah. Lakukan hal yang sama ketika $d(X,u)+d(X,v) > d(Y,u)+d(Y,v)$.

Teorema. *Dalam sebuah poligon reguler ukuran N dengan* $2N-3$ *edges, terdapat sebuah edge* (X,Y) *bukan sisi yang membagi poligon menjadi dua poligon konveks ukuran L dan R dengan* $\max(L,R) \leq 1+\frac{2N}{3}$.

Bukti. Anggap kita memiliki poligon reguler dengan ukuran N. Gambar sebuah lingkaran sehingga semua verteks pada poligon berada di lingkaran. Perhatikan titik pusat dari lingkaran tersebut.

Jika titik pusat dari lingkaran berada di perbatasan dua segitiga, maka garis yang memuat titik pusat merupakan diameter dari lingkaran. Sehingga, garis ini membagi poligon ukuran N menjadi dua buah poligon dengan ukuran $\frac{N+2}{2}$. $\frac{N+2}{2} \leq 1+\frac{2N}{3} \rightarrow 0 \leq N$, maka kita selesai.

Sekarang kita dapat berasumsi titik pusat dari lingkaran berada di dalam sebuah segitiga. Perhatikan karena verteks-verteks segitiga berada pada lingkaran, berarti segitiga ini membagi poligon ukuran N menjadi empat buah poligon dengan ukuran S, T, U, dan 3 (segitiga itu sendiri). Anggap $P = N-2$ adalah banyaknya segitiga dalam poligon dengan N sisi. Karena segitiga memuat titik pusat lingkaran, $\frac{P}{3} \leq \max(S-2, T-2, U-2) \leq \frac{P-1}{2}$ berlaku. Maka salah satu sisi segitiga tersebut membagi poligon menjadi dua bagian sehingga $\max(L-2, R-2) \leq P-\frac{P-1}{3}$. Subsitusi dengan $P = N-2$, maka kita mendapatkan $\max(L,R)-2 \leq (N-1)-\frac{N}{3} \rightarrow \max(L,R) \leq 1+\frac{2N}{3}$. □

Jika kita memilih edge pembagi (X,Y) yang meminimalkan $\max(|L|,|R|)$, kompleksitas dari solusi ini adalah $O(N\log^2 N)$ karena teorema di atas.

Kompleksitas waktu: $O(N\log^2 N)$

	Title	Author	Preparation	Editorialist
A	Puzzling Puzzle	Pyqe	Pyqe	rama_pang
B	Eating Education	Berted	Berted	Berted
C	Calculating Cuteness	Pyqe	Pyqe	rama_pang
D	Compassionate Companions	Berted	Berted	rama_pang
E	Multiple Multiplication	rama_pang	prabowo	rama_pang
F	Radio Radius	dascogabriel	dascogabriel	dascogabriel
G	City Circle	rama_pang	rama_pang	rama_pang

A. Puzzling Puzzle

Observe that a rectangle with sides 1 and N has an area of $1 \cdot N = N$. Simply output 1 and N in this case.

Time Complexity: $O(1)$

B. Eating Education

For the rest of the discussion, we assume A is sorted and 0-indexed.

The minimum time for Djodjo to move from x to y clockwise is $y-x$ if $y > x$ or $M-y+x$ if $y < x$.

Observation. *In an optimal movement sequence, Djodjo will move from A_i to $A_{(i-1) \bmod N}$ clockwise for some index i.*

We can simply compute the time needed for every possible i and output the minimum of them.

Time complexity: $O(N \log N)$

C. Calculating Cuteness

Observation. *Array A is not cute if there is no element with value M.*

Observation. *Array A is not cute if there are two elements with value M next to each other.*

Proof. If there exist two adjacent elements with value M, then both of these elements cannot be erased. Meanwhile, an array is cute only if we can erase all elements except one with value M. □

Observation. *All arrays A not satisfying the two conditions above is cute.*

Proof. While there exist more than 1 element with value M, erase that element. Since there are no two adjacent elements with value M, this is always possible. Then, we can erase the elements to the left and right of the single remaining element with value M since all other elements have less values. □

We can use dynamic programming to find the number of arrays which does not have two elements with value M next to each other. Let $dp[n][0]$ be the number of arrays with length n such that the n-th element does not have value M, and $dp[n][1]$ be the number of arrays with length n such that the n-th element has value M.

The base case for dp is $dp[0][0] = 1$ and $dp[0][1] = 0$.

$dp[n][0] = (M-1) \cdot (dp[n-1][0] + dp[n-1][1])$, since the n-th element is not M we can append it to all valid arrays with length $n-1$.

$dp[n][1] = dp[n-1][0]$, since if we append the value M then the $(n-1)$-th element cannot be M.

Thus the number of arrays with length N such that no two elements with value M are adjacent is $dp[N][0] + dp[N][1]$. However, an array is cute if there exists at least one element with value M. Thus we have to decrease the answer by the number of arrays with no value M, which equals $(M-1)^N$.

Thus the number of cute arrays is $dp[N][0] + dp[N][1] - (M-1)^N$. dp has $O(N)$ states and for each state, the transition can be done in $O(1)$. Therefore, dp can be computed in $O(N)$.

Time complexity: $O(N)$

D. Compassionate Companions

We will binary search the answer. Assume we are checking whether $answer \geq mid$. Create an edge (u, v) if $A[u][v] < mid$, and let this graph be graph G. Observe that if two vertices are connected by an edge in G, then that means both of these vertices must belong to a different group.

Thus for each vertex u, we need to color it in one of two possible colors such that no two adjacent vertices have the same color. This is equivalent to checking whether G is a bipartite graph.

Observe that in the binary search, there are only $O(N^2)$ possible candidates to the answer. We can binary search only on these $O(N^2)$ values.

Time complexity: $O(N^2 \log N)$

E. Multiple Multiplication

Definition. *$f(i, j)$ is the lowest common multiple of $A[i], A[i+1], \ldots, A[j]$.*

Definition. *$cnt_p[i]$ is the maximum possible x such that $p^x \mid A[i]$.*

Definition. *$rangemax_p(i, j)$ is the maximum value of $cnt_p[i], cnt_p[i+1], .., cnt_p[j]$.*

Theorem. $\prod_{i=1}^{N} \prod_{j=i}^{N} f(i,j) = \prod_{p \in prime} p^{\sum_{i=1}^{N} \sum_{j=i}^{N} rangemax_p(i,j)}$

For all primes p, we can compute $\sum_{i=1}^{N} \sum_{j=i}^{N} rangemax_p(i, j)$ in $O(N)$ or $O(N \log A_{max})$ using a stack. Iterate r from 1 to N while maintaining a stack containing pairs (mx, id), where $stack[i].mx \geq stack[i+1].mx$ and id is the greatest index l such that $rangemax_p(i, r) \geq mx$ for $1 \leq i \leq l$.

Observation. *An integer x has $O(\log x)$ prime factors.*

Corollary. $rangemax_p(i,r) \leq O(\log A_{max})$.

Corollary. $\sum_{p \in prime} \sum_{i=1}^{N} cnt_p[i] = O(N \log A_{max})$.

Note that we can compute $\sum_{i=1}^{r} rangemax_p(i,r)$ by iterating all values in the stack. Since the number of elements in the stack is at most $O(\log A_{max})$, this algorithm runs in $O(N \log A_{max})$. To reduce it to $O(N)$, we can maintain another variable $power\_sum = \sum_{i=1}^{r} rangemax_p(i,r)$.

When updating from r to $r+1$, we can simply pop all elements in the stack while $stack.top().mx \leq cnt_p[r]$, then push the pair $(cnt_p[r], r)$ into the stack. In the $O(N)$ solution, we will also update *power_sum* every time we pop or push the stack.

The algorithm above can be optimized further by not processing all indices i where $cnt_p[i] = 0$. Since $\sum_{p \in prime} \sum_{i=1}^{N} cnt_p[i] = O(N \log A_{max})$, the complexity of this solution is either $O(N \log A_{max})$ or $O(N \log^2 A_{max})$ depending on the implementation of the algorithm.

There also exist other solutions without using that particular optimization.

Observation. *If $p > \sqrt{A_{max}}$, then $0 \leq cnt_p[i] \leq 1$.*

Theorem. *Assume $\pi(x)$ is a function which returns the number of primes $p \leq x$, then $\pi(x) = O(\frac{x}{\log x})$.*

We will use the algorithm above for all primes $p \leq \sqrt{A_{max}}$. Since there are $O(\frac{\sqrt{A_{max}}}{\log A_{max}})$ primes $p \leq \sqrt{A_{max}}$, the complexity for this part is $O(N \frac{\sqrt{A_{max}}}{\log A_{max}})$ or $O(N\sqrt{A_{max}})$. For $p > \sqrt{A_{max}}$, we simply need to count how many subarrays have an element 1 in it. This can be done easily in $O(\sum_{i=1}^{N} cnt_p[i])$.

Time complexity: $O((N + A_{max}) \log A_{max})$

F. Radio Radius

Definition. *Radio tower j belongs to mask if and only if the j-th bit of mask is on.*

Definition. *$f(mask)$ is the number of houses that can be reached by **all** radio towers that belong to mask*

Definition. *$g(mask)$ is the number of houses that can be reached by **at least one** radio tower that belongs to mask.*

Definition. *A house is perfectly reachable by a set of radio towers S if and only if every radio tower in S can reach the house.*

Observe that if a house is perfectly reachable by a set of radio towers S, then the house is also perfectly reachable by all subsets of S. For each house i, we find the maximal set of radio towers

S and store into an array H. Then, we can calculate $f(mask)$ by using sum over superset dynamic programming on the array H.

Afterward, we can calculate the value of $g(mask)$ from the values of $f(mask)$ by using sum over subset dynamic programming and the principle of inclusion and exclusion.

Time complexity: $O(NM + M2^M)$

G. City Circle

Before we begin, we can change an arc edge W_i into a straight line edge $(i, i+1)$ with weight W_i. If there are duplicates, simply take the one with minimum weight.

Definition. *A polygon with size N is a polygon with N vertices and N sides.*

Definition. *A non-side edge of a polygon is a straight line connecting two vertices of the polygon which are not adjacent.*

The initial graph given is a polygon with size N and M edges. We consider both side edges and non-side edges.

Lemma. *For a convex polygon with N sides and M non-side edges which do not intersect each other, there exists an isomorphic regular polygon with N sides and M non-side edges which do not intersect each other.*

Lemma. *We can partition a regular polygon with size N into $N-2$ triangles by doing the following operation while possible: choose 2 vertices and connect them with a straight line if it does not intersect with any other existing lines. For any possible configuration and order of operations, we will end up with $N-2$ triangles.*

Proof. We will prove it through induction. Let $f(N)$ be the number of triangles for a regular polygon with size N.

For $N = 3$, then the polygon is also a triangle, which satisfies $f(3) = 3-2 = 1$.

For $N > 3$, assume $f(x) = x-2$ for all $x < N$. There exists at least one non-side edge which splits the polygon into two smaller convex polygons with sizes L and R. Then $L+R = N+2$ since the splitting edge exist on both polygons, and $f(N) = f(L) + f(R)$. Since $3 \leq L, R$, then $L, R < N$. Therefore $f(N) = (L-2) + (R-2) = L+R-4$. Substitute $L+R = N+2$ to get $f(N) = N+2-4 = N-2$. □

Theorem. *$M \leq 2N-3$ and for any configuration of the M edges M, we can add $2N-3-M$ additional edges with weight ∞, such that there exists a total of $2N-3$ edges on a polygon with size N. This does not affect the answer.*

Proof. Since all additional edges will have weight ∞, it will never be used in an optimal path. Assume we have a regular polygon with size N. Each vertex is numbered from 1 to N clockwise. For every edge (u, v), draw a straight line from u to v.

Observe that the polygon with size N is in the process of being partitioned into $N-2$ triangles. Since there will finally be $N-2$ triangles, that means the polygon will be split by $N-3$ non-side edges. There are N side edges and $N-3$ non-side edges in total. Out of the $2N-3$ edges, there already exist M edges. Thus to complete the partition, we have to add another $2N-3-M$ edges. □

Definition. *$path(u,v)$ is the optimal path from u to v.*

Definition. *$d(u,v)$ is the total length of the optimal path from u to v.*

We can now assume $M = 2N-3$ by adding additional edges with weight ∞. To add those edges, assume we initially have a set of vertices $S = \{1,2,..,N\}$. Find a vertex $u \in S$ with degree 2 and connect the vertices adjacent to it by an edge, then erase u from S. Repeat while $|S| > 3$.

We will now count $\sum_{u=1}^{N-1}\sum_{v=u+1}^{N} d(u,v)$ with a divide and conquer approach. $N = 3$ can be handled easily. We can now assume $N > 3$.

In a regular polygon with edges, we can choose a non-side edge and split the polygon into two smaller polygons. Assume the chosen edge is (X,Y). L and R are the set of vertices of the two smaller polygons.

There are a few problems before we can recurse to L and R. Note that $path(u,v)$ where $u,v \in L$ can go through some vertex $w \in R$. However, observe that $path(u,v)$ will go through X and Y in those cases. Therefore, we can run Dijkstra's algorithm to determine the optimal distance from X to Y then change the weight of (X,Y) accordingly. Now both L and R are independent of each other.

To merge L and R, we have to count the sum of $d(u,v)$ for $u \in L$ and $v \in R$. $path(u,v)$ must go through either X or Y, thus $d(u,v) = \min(d(X,u)+d(X,v), d(Y,u)+d(Y,v))$. We will count the sum of $d(u,v)$ when $d(X,u)+d(X,v) \leq d(Y,u)+d(Y,v)$.

$d(X,u)+d(X,v) \leq d(Y,u)+d(Y,v)$ can be rewritten as $d(X,u)-d(Y,u) \leq d(Y,v)-d(X,v)$. Sort the vertices in R based on $d(Y,v)-d(X,v)$. Then, for every vertex $u \in L$, we can count the sum of $d(X,v)$ for all $v \in R$ satisfying $d(X,u)-d(Y,u) \leq d(Y,v)-d(X,v)$ easily. We can use the same approach for $d(X,u)+d(X,v) > d(Y,u)+d(Y,v)$.

Theorem. *In a regular polygon with N vertices and $2N-3$ edges, there exists a non-side edge (X,Y) which splits the polygon into two convex polygons with sizes L and R, such that* $\max(L,R) \leq 1+\frac{2N}{3}$.

Proof. Assume we have a regular polygon with size N. Draw a circle such that all vertices of the polygon is on the circle. Observe the origin of the circle.

If the origin of the circle is on a line between two triangles, then that particular line is the diameter of the circle. That means this line splits the polygon with size N into two polygons with size $\frac{N+2}{2}$. Since $\frac{N+2}{2} \leq 1+\frac{2N}{3} \to 0 \leq N$, we are done.

Now we can assume the origin of the circle is in a triangle. Observe that since the vertices of the triangle are on the circle, the polygon with size N is divided into four polygons with sizes S, T, U, and 3 (the triangle itself). Assume $P = N-2$ is the number of triangles in a polygon with size N. Since the triangle includes the origin of the circle, $\frac{P-1}{3} \leq \max(S-2,T-2,U-2) \leq \frac{P}{2}$ holds.

That means one of the sides of the triangle divides the polygon into two polygons with sizes L and R such that $\max(L-2,R-2) \leq P - \frac{P-1}{3}$. Substitute $P = N-2$, then we have $\max(L,R) - 2 \leq (N-1) - \frac{N}{3} \rightarrow \max(L,R) \leq 1 + \frac{2N}{3}$. □

If we choose a splitting edge (X,Y) which minimizes $\max(|L|,|R|)$, the complexity of the solution is $O(N \log^2 N)$ due to the previous theorem.

Time complexity: $O(N \log^2 N)$