國立臺南大學資訊工程學系

資工三「演算法」課程

第三次作業

**題目: Electirc Vehicle Moter 2**

|  |  |  |
| --- | --- | --- |
| 班級 | ： | 資工三 |
| 姓名 | ： | 周巧晴 |
| 學號 | ： | S10959042 |

老師：陳宗禧

中華民國 111年11月28日

# 目錄

1. **簡介及問題描述摘要….……………..…………………………………………3**
   1. **簡介…………….…………………………………………………………………………3**
   2. **問題………………………….……………………………………………………………3**
2. **理論分析….………………………..………………………………………4**
3. **演算法則….…………………………..……………………………………6**
4. **程式設計環境架構.………………………..…………………………………47**
5. **程式.…………………………………………..………………………………48**
6. **執行結果、討論與心得.………………………..……………………………270**

參考文獻…………………………………………………………………………285

**(一) 簡介及問題描述摘要**

1. 簡介

給定一地圖，設有*n*個u-motor充電站(還電動機車與充電站)，例：充電站設置在台南古蹟或車站旁等。底下有幾個問題要解決：

2. 問題

(1) 依據充電站的位置(2D)，建構一個Voronoi Diagram with time complexity O(𝑛log𝑛).

(2) 依序列出充電站的服務範圍(面積)。

(3) 找出離充電站最遠的點以及距離(Largest Empty Circle Problem with Voronoi Diagram, also called toxic waste dump problem)。

(4) 根據Sub-problem (3)，在該位置安裝一個充電站，則須在原來的Voronoi Diagram外，加一個點變成有*n+1*個點的Voronoi Diagram，設計一程式法建構該新的Voronoi Diagram。

(5) 重複Sub-problems (3) and (4)直到最遠距離滿足一個門檻值(Threshold value *k*, 如500m，*k*值可自行設定)，並輸出總共需要幾個充電站、目前離充電站最遠的距離與*k*值。

**Notes:**

(1) 輸入(Input) n個二維的座標充電站。

(2) 請自行定義空間距離的單位，如公尺m。

**(二) 理論分析**

1. fortune Algorithm

(1) 依據充電站的位置(2D)，建構一個Voronoi Diagram with time complexity O(𝑛log𝑛).

Input：一張a\*b具有n個充電站的地圖(單位公里)，x=0代表空的，x>0代表充電站。

先找一遍地圖，找出所有充電站的座標並記錄下來。

用一條掃描線從y=0往y大的方向掃描，遇到充電站點則產生點事件，將以此點為焦點，掃描線為準線的拋物線放入beachline，兩相鄰拋物線的斷點為edge，三相鄰拋物線的斷點為vertex，同時計算vertex事件和edge事件，並記錄發生時的y，當產生vertex事件會刪掉中間的拋物線，當edge碰到邊界，會產生邊界的vertex事件，每處理完一個事件就判斷下一個事件是誰(y最小)，直到所有點都處理完、事件也處理完，則結束，再加入(0，0)、(0，b)、(a，0)、(a，b)點。

Time Complexity?

找一遍地圖是O(a\*b)，做fortune algorithm是O(nlogn)。

所以是O(a\*b)+O(nlogn)。

(2) 依序列出充電站的服務範圍(面積)。

Input：一張a\*b具有n個充電站的地圖(單位公里)，x=0代表空的，x>0代表充電站。

先找一遍地圖，找出所有充電站的座標並記錄下來。

做完fortune後將每個充電站產生的vertex(還有判斷(0，0)、(0，b)、(a，0)、(a，b)是否離此充電站最近)作為凸多邊形頂點，逆時針排序vertex後，用面積=公式，算出面積。

Time Complexity?

找一遍地圖是O(a\*b)，做fortune是O(nlogn)，排序為逆時針是O(n3)，計算面積是O(n2)，排序面積是O(n2)。

所以O(a\*b)+O(nlogn)+ O(n3)+ O(n2)+ O(n2)= O(a\*b)+O(n3)。

(3) 找出離充電站最遠的點以及距離(Largest Empty Circle Problem with Voronoi Diagram, also called toxic waste dump problem)。

Input：一張a\*b具有n個充電站的地圖(單位公里)，x=0代表空的，x>0代表充電站。

先找一遍地圖，找出所有充電站的座標並記錄下來。

做完fortune後，計算每個vertex(邊界的vertex也算)和產生它的拋物線的焦點(充電站)之間的距離，以及四個角(0，0)、(0，b)、(a，0)、(a，b)與離它最近的充電站之間的距離，並找出最大值，將此點的座標和與充電站的距離印出。

Time Complexity?

找一遍地圖是O(a\*b)，做fortune是O(nlogn)，計算vertex是O(n)，計算四個角是O(n)。

所以是O(a\*b)+O(nlogn)+O(n)+O(n)= O(a\*b)+O(nlogn)。

(4) 根據Sub-problem (3)，在該位置安裝一個充電站，則須在原來的Voronoi Diagram外，加一個點變成有*n+1*個點的Voronoi Diagram，設計一程式法建構該新的Voronoi Diagram。

Input：一張a\*b具有n個充電站的地圖(單位公里)，x=0代表空的，x>0代表充電站。

先找一遍地圖，找出所有充電站的座標並記錄下來。

做完fortune後，計算每個vertex(邊界的vertex也算)和產生它的拋物線的焦點(充電站)之間的距離，以及四個角(0，0)、(0，b)、(a，0)、(a，b)與離它最近的充電站之間的距離，並找出最大值，將此點納入充電站，並且把此點和產生它的拋物線的焦點加入要討論的充電站點中，並判斷是否有vertex(或邊界的vertex)與產生它的拋物線的焦點的距離大於這些vertex(或邊界的vertex)與新點的距離，有的話就將這些vertex(或邊界的vertex)刪掉，並把產生這些vertex(或邊界的vertex)的拋物線全部重新做fortune，最後再把產生的錯誤的vertex和邊界的vertex刪掉。

Time Complexity?

找一遍地圖是O(a\*b)，做fortune是O(nlogn)，計算vertex是O(n)，計算四個角是O(n)，判斷是否有vertex與產生它的拋物線的焦點的距離大於這些vertex與新點的距離是O(n)。

所以O(a\*b)+O(nlogn)+O(n)+O(n)+O(n)= O(a\*b)+O(nlogn)。

(5) 重複Sub-problems (3) and (4)直到最遠距離滿足一個門檻值(Threshold value *k*, 如500m，*k*值可自行設定)，並輸出總共需要幾個充電站、目前離充電站最遠的距離與*k*值。

Input：一張a\*b具有n個充電站的地圖(單位公里)，x=0代表空的，x>0代表充電站。

先找一遍地圖，找出所有充電站的座標並記錄下來。

做完fortune後不斷重複(3)、(4)題直到最遠距離<=門檻值。

Time Complexity?

找一遍地圖是O(a\*b)，做fortune是O(nlogn)，最多會重複O(n2/k2)次，所以是O(a\*b)+O(n3logn/k2)。

**(三) 演算法則**

1. Fortune(Algorithm)

(1) 依據充電站的位置(2D)，建構一個Voronoi Diagram with time complexity O(𝑛log𝑛).

Input：一張a\*b具有n個充電站的地圖(單位公里)，x=0代表空的，x>0代表充電站。

Output：所有產生的vertex點和邊界的vertex點座標。

index=0

for i=0 to a do

for j=0 to b do

if(map[i][j]>0)

p[index].x=j

p[index++].y=i

index2=0,index3=0,index4=0,index5=0,index6=0

while(index!=0)

if(index3!=0&&etop(ev)->y<=ptop(p).y&&etop(ev)->y<=betop(bev)->y)

process\_event(ev,&index3,vt,&index4,bev,&index5,bvt,&index6)

else if(index5==0|| ptop(p).y<= betop(bev)->y)

process\_point(p,&index,&beachline,ev,&index3,bev,&index5)

else if(index5!=0)

process\_bevent(bev,&index5,bvt,&index6)

while(index3!=0||index5!=0)

if(index5==0||(index3!=0&&etop(ev)->y<=betop(bev)->y)

process\_event(ev,&index3,vt,&index4,bev,&index5,bvt,&index6)

else if(index5!=0)

process\_bevent(bev,&index5,bvt,&index6)

freeb(&beachline)

printf(“點為:\n”)

for i=0 to index4-1 do

printf(“(%.2f,&.2f)\n”,vt[i].p.x,vt[i].p.y)

for i=0 to index6-1 do

printf(“(%.2f,&.2f)\n”,bvt[i].p.x,bvt[i].p.y)

print(bvt,index6,vt,index4)

//為了釋放掉還沒釋放的beachline的空間

//Input：beachline(\*b)

//Output：NONE

freeb(\*b) {

if (\*b == NULL)

return

freeb(&((\*b)->rp))

free(\*b)

//印出四個角的值

//Input：邊界的vertex(bvt)和vertex(vt)以及他們的長度(index6,index4)

//Output：NONE

print(bvt, index6, vt, index4) {

ife = true

for i = 0 to index6-1 do

if (bvt[i].p.x == 0 && bvt[i].p.y == 0)

ife = false

for i = 0 to index4 do

if (vt[i].p.x == 0 && vt[i].p.y == 0)

ife = false

if (ife)

printf("(0.00,0.00)\n")

ife = true

for i = 0 to index6-1 do

if (bvt[i].p.x == 0 && bvt[i].p.y == maxy)

ife = false

for i = 0 to index4-1 do

if (vt[i].p.x == 0 && vt[i].p.y == maxy)

ife = false

if (ife)

printf("(0.00,%d.00)\n", maxy)

ife = true

for i = 0 to index6-1 do

if (bvt[i].p.x == maxx && bvt[i].p.y == 0)

ife = false

for i = 0 to index4-1 do

if (vt[i].p.x == maxx && vt[i].p.y == 0)

ife = false

if (ife)

printf("(%d.00,0.00)\n", maxx)

ife = true

for i = 0 to index6-1 do

if (bvt[i].p.x == maxx && bvt[i].p.y == maxy)

ife = false

for i = 0 to index4-1 do

if (vt[i].p.x == maxx && vt[i].p.y == maxy)

ife = false

if (ife)

printf("(%d.00,%d.00)\n", maxx, maxy)

//將充電站點從佇列彈出

//Input：要彈出的充電站佇列(\*p)和它的長度(\*index)

//Output：彈出的充電站點值

ppop(\*p, \*index)

p0 = p[0]

for i = 0 to \*index – 2 do

p[i] = p[i + 1]

(\*index)--

return p0

//提供下一次ppop會彈出的充電站點值

//Input：充電站佇列(\*p)

//Output：下一次ppop會彈出的充電站點值

ptop(\* p)

return p[0]

//將vertex事件從佇列彈出

//Input：要彈出的vertex事件佇列(\*\* ev)和它的長度(\*index)

//Output：彈出的vertex事件

epop(\*\* ev, \* index)

e0 = ev[0]

for i = 0 to \*index – 2 do

ev[i] = ev[i + 1]

(\*index)--

return e0

//提供下一次epop會彈出的vertex事件值

//Input：vertex事件佇列(\*\*ev)

//Output：下一次epop會彈出的vertex事件值

etop(\*\* ev)

return ev[0]

//將邊界的vertex事件從佇列彈出

//Input：要彈出的邊界的vertex事件佇列(\*\* bev)和它的長度(\*index)

//Output：彈出的邊界的vertex事件

bepop(\*\* bev, \* index)

\*be0 = bev[0]

for i = 0 to \*index – 2 do

bev[i] = bev[i + 1]

(\*index)--

return be0

//提供下一次bepop會彈出的邊界的vertex事件值

//Input：vertex事件佇列(\*\*ev)

//Output：下一次bepop會彈出的邊界的vertex事件值

betop(\*\* bev)

return bev[0]

//刪除掉某個beachline拋物線

//Input：要刪除的beachline拋物線

//Output：NONE

d(b)

if (b->bev)

b->bev->valid = false

b->bev = NULL

if (b->lp->bev)

b->lp->bev->valid = false

b->lp->bev = NULL

if (b->bevd)

b->bevd->valid = false

b->bevd = NULL

if (b->lp->bevd)

b->lp->bevd->valid = false

b->lp->bevd = NULL

if (b->bevr)

b->bevr->valid = false

b->bevr = NULL

if (b->lp->bevr)

b->lp->bevr->valid = false

b->lp->bevr = NULL

if (b->bevl)

b->bevl->valid = false

b->bevl = NULL

if (b->lp->bevl)

b->lp->bevl->valid = false

b->lp->bevl = NULL

if (b->ev)

b->ev->valid = false

b->ev = NULL

if (b->lp->ev)

b->lp->ev->valid = false

b->lp->ev = NULL

if (b->rp->ev)

b->rp->ev->valid = false

b->rp->ev = NULL

if (b->rp && b->lp)

b->rp->lp = b->lp

b->lp->rp = b->rp

else if (b->rp)

b->rp->lp = NULL

else if (b->lp)

b->lp->rp = NULL

//先遇到充電站點處理充電站點事件，將點放入充電站點佇列

//Input：充電站點佇列(\* p)和它的長度(\* index)、整條beachline(\* b)、vertex事件佇列(\*\* ev)和它的長度(\* index3)邊界的vertex事件佇列(\*\* bev)和它的長度(\* index5)

//Output：NONE

process\_point(\* p, \* index, \* b, \*\* ev, \* index3, \*\* bev, \* index5)

p0 = ppop(p, index)

insert(p0, b, ev, index3, bev, index5)

//先遇到vetex事件處理vertex事件，將有效的vertex產生，並刪掉中間的beachline

// Input：vertex事件佇列(\*\* ev)和它的長度(\* index3)、vertex陣列(\*v)和它的長度(\*index4)、邊界的vertex事件佇列(\*\* bequeue)和它的長度(\* index5) 、邊界的vertex陣列(\*bvt)和它的長度(\*index6)

//Output：NONE

process\_event(\*\* equeue, \* index3, \* v,\* index4, \*\* bequeue, \* index5, \* bvt, \* index6) {

\* ev = epop(equeue, index3)

if (ev->valid)

if (ev->o.x > maxx || ev->o.x < 0 || ev->o.y > maxy || ev->o.y < 0)

else

v[\*index4].p = ev->o

v[\*index4].p2 = ev->b->p

v[\*index4].p1 = ev->b->lp->p

v[(\*index4)++].p3 = ev->b->rp->p

d(ev->b)

boundu\_event(&ev->b->lp, bequeue, index5, ev->y)

boundd\_event(&ev->b->lp, bequeue, index5, ev->y)

boundr\_event(&ev->b->lp, bequeue, index5, ev->y)

boundl\_event(&ev->b->lp, bequeue, index5, ev->y)

circle\_event(&ev->b->lp, ev->y, equeue, index3)

circle\_event(&ev->b->rp, ev->y, equeue, index3)

if (ev->b->lp)

free(ev->b)

else if (!ev->b->lp && ev->b->rp)

ev->b = ev->b->rp

else

ev->b = NULL

free(ev)

//先遇到邊界的vetex事件處理邊界的vertex事件

// Input：邊界的vertex事件佇列(\*\* bequeue)和它的長度(\* index5) 、邊界的vertex陣列(\*bvt)和它的長度(\*index6)

//Output：NONE

process\_bevent(\*\* bequeue, \* index5, \* bv, \* index6)

\* bev = bepop(bequeue, index5)

if (bev->valid)

bv[\*index6].p = bev->o

bv[\*index6].p2 = bev->b1->p

bv[(\*index6)++].p1 = bev->b2->p

if (bev->o.y == 0)

bev->b1->bev = NULL

else if (bev->o.y == maxy)

bev->b1->bevd = NULL

else if (bev->o.x == 0)

bev->b1->bevl = NULL

else if (bev->o.x == maxx)

bev->b1->bevr = NULL

free(bev)

//判斷是否有上邊界的vertex事件發生，並記錄它發生時的掃描線y及邊界的vertex點座標和產生它的兩個拋物線

//Input： 整條beachline(\* b)、邊界的vertex事件佇列(\*\* bequeue)和它的長度(\* index5)、當下的掃描線(y0)

//Output：NONE

boundu\_event(\* b, \*\* bequeue, \* index5, y0)

if (!\*b || !(\*b)->rp)

return

if (boundu((\*b)->p, (\*b)->rp->p, &y, &o) && y >= y0)

if (o.x <= (\*b)->p.x)

if (!(\*b)->lp || intersection((\*b)->lp->p, (\*b)->rp->p, y).y < o.y)

else

return

else if (o.x >= (\*b)->rp->p.x)

if (!(\*b)->rp->rp || intersection((\*b)->p, (\*b)->rp->rp->p, y).y < o.y)

else

return;

bev = (be\*)malloc(sizeof(\*bev))

bev->b1 = \*b

bev->b2 = (\*b)->rp

bev->o = o

bev->valid = true

bev->y = y

(\*b)->bev = bev

insertbe(bequeue, bev, index5)

//判斷是否有下邊界的vertex事件發生，並記錄它發生時的掃描線y及邊界的vertex點座標和產生它的兩個拋物線

//Input： 整條beachline(\* b)、邊界的vertex事件佇列(\*\* bequeue)和它的長度(\* index5)、當下的掃描線(y0)

//Output：NONE

boundd\_event(\* b, \*\* bequeue, \* index5, y0) {

if (!\*b || !(\*b)->rp)

return

if (boundd((\*b)->p, (\*b)->rp->p, &y, &o) && y >= y0)

bev = (be\*)malloc(sizeof(\*bev))

bev->b1 = \*b

bev->b2 = (\*b)->rp

bev->o = o

bev->valid = true

bev->y = y

(\*b)->bevd = bev

insertbe(bequeue, bev, index5)

//判斷是否有右邊界的vertex事件發生，並記錄它發生時的掃描線y及邊界的vertex點座標和產生它的兩個拋物線

//Input： 整條beachline(\* b)、邊界的vertex事件佇列(\*\* bequeue)和它的長度(\* index5)、當下的掃描線(y0)

//Output：NONE

boundr\_event(\* b, \*\* bequeue, \* index5, y0)

if (!\*b || !(\*b)->rp)

return

if (boundr((\*b)->p, (\*b)->rp->p, &y, &o) && y >= y0)

bev = (be\*)malloc(sizeof(\*bev))

bev->b1 = \*b

bev->b2 = (\*b)->rp

bev->o = o

bev->valid = true

bev->y = y

(\*b)->bevr = bev

insertbe(bequeue, bev, index5)

//判斷是否有左邊界的vertex事件發生，並記錄它發生時的掃描線y及邊界的vertex點座標和產生它的兩個拋物線

//Input： 整條beachline(\* b)、邊界的vertex事件佇列(\*\* bequeue)和它的長度(\* index5)、當下的掃描線(y0)

//Output：NONE

void boundl\_event(\* b, \*\* bequeue, \* index5, y0)

if (!\*b || !(\*b)->rp)

return

if (boundl((\*b)->p, (\*b)->rp->p, &y, &o) && y >= y0)

bev = (be\*)malloc(sizeof(\*bev))

bev->b1 = \*b

bev->b2 = (\*b)->rp

bev->o = o

bev->valid = true

bev->y = y

(\*b)->bevl = bev

insertbe(bequeue, bev, index5)

//判斷兩拋物線是否有斷點在範圍內的左邊界上，並產生發生時的掃描線y及邊界的vertex點座標

//Input：x比較小的拋物線的焦點(p1)，x比較大的拋物線的焦點(p2)，用於紀錄產生的發生時的掃描線y(\*y)，用於紀錄產生的邊界的vertex點座標(\*o)

//Output：有斷點在範圍內的左邊界上則true，沒斷點在範圍內的左邊界上則false

boundl(p1, p2, \* y, \* o)

if (p1.y >= p2.y)

return false

oy = ((p1.x \* p1.x + p1.y \* p1.y) - (p2.x \* p2.x + p2.y \* p2.y)) / (2 \* (p1.y - p2.y))

if (oy < maxy && oy > 0)

double y0 = (2 \* oy + sqrt(4 \* oy \* oy - 4 \* (2 \* oy \* p1.y - p1.x \* p1.x - p1.y \* p1.y))) / 2

o->x = 0

o->y = oy

\*y = y0

return true

return false

//判斷兩拋物線是否有斷點在範圍內的右邊界上，並產生發生時的掃描線y及邊界的vertex點座標

//Input：x比較小的拋物線的焦點(p1)，x比較大的拋物線的焦點(p2)，用於紀錄產生的發生時的掃描線y(\*y)，用於紀錄產生的邊界的vertex點座標(\*o)

//Output：有斷點在範圍內的右邊界上則true，沒斷點在範圍內的右邊界上則false

bool boundr(p1, p2, \* y, \* o) {

if (p1.y <= p2.y)

return false

oy = (2 \* maxx \* (p2.x - p1.x) + p1.x \* p1.x + p1.y \* p1.y - p2.x \* p2.x - p2.y \* p2.y) / (2 \* (p1.y - p2.y))

if (oy < maxy && oy > 0)

y0 = (2 \* oy + sqrt(4 \* oy \* oy - 4 \* (2 \* oy \* p1.y + 2 \* p1.x \* maxx - maxx \* maxx - p1.x \* p1.x - p1.y \* p1.y))) / 2

o->x = maxx

o->y = oy

\*y = y0

return true

return false

//判斷兩拋物線是否有斷點在範圍內的下邊界上，並產生發生時的掃描線y及邊界的vertex點座標

//Input：x比較小的拋物線的焦點(p1)，x比較大的拋物線的焦點(p2)，用於紀錄產生的發生時的掃描線y(\*y)，用於紀錄產生的邊界的vertex點座標(\*o)

//Output：有斷點在範圍內的下邊界上則true，沒斷點在範圍內的下邊界上則false

bool boundd(p1, p2, \* y, \* o)

if (p1.x == p2.x)

return false

ox = (2 \* maxy \* (p2.y - p1.y) + p1.x \* p1.x + p1.y \* p1.y - p2.x \* p2.x - p2.y \* p2.y) / (2 \* (p1.x - p2.x))

if (ox <= maxy && ox >= 0)

y0 = (2 \* maxy + sqrt(4 \* maxy \* maxy - 4 \* (2 \* ox \* p1.x + 2 \* p1.y \* maxy - ox \* ox - p1.x \* p1.x - p1.y \* p1.y))) / 2

o->x = ox

o->y = maxy

\*y = y0

return true

return false

//判斷兩拋物線是否有斷點在範圍內的上邊界上，並產生發生時的掃描線y及邊界的vertex點座標

//Input：x比較小的拋物線的焦點(p1)，x比較大的拋物線的焦點(p2)，用於紀錄產生的發生時的掃描線y(\*y)，用於紀錄產生的邊界的vertex點座標(\*o)

//Output：有斷點在範圍內的上邊界上則true，沒斷點在範圍內的上邊界上則false

boundu(p1, p2, \* y, \* o)

if (p1.x == p2.x)

return false

ox = ((p1.x \* p1.x + p1.y \* p1.y) - (p2.x \* p2.x + p2.y \* p2.y)) / (2 \* (p1.x - p2.x))

if (ox <= maxx && ox >= 0)

double y0 = sqrt(ox \* ox - 2 \* ox \* p1.x + p1.x \* p1.x + p1.y \* p1.y)

o->x = ox

o->y = 0

\*y = y0

return true

return false

//將新的vertex事件依發生時的掃描線y排列進vertex事件佇列裡

//Input：vertex事件佇列(\*\*equeue)和它的長度(\*index3)、要插入的vertex事件(\*ev)

//Output：NONE

inserte(\*\* equeue, \* ev, \* index3)

if (\*index3 == 0)

equeue[0] = ev

(\*index3)++

return

if (equeue[0]->y >= ev->y)

for j = \*index3 downto 1 do

equeue[j] = equeue[j - 1]

equeue[0] = ev

(\*index3)++

return

if (equeue[\*index3 - 1]->y <= ev->y)

equeue[\*index3] = ev

(\*index3)++

return

for i = 0 to \*index3 – 2 do

if (equeue[i]->y <= ev->y && equeue[i + 1]->y >= ev->y)

for j = \*index3 downto i + 2 do

equeue[j] = equeue[j - 1]

equeue[i + 1] = ev

(\*index3)++

Break

//將新的邊界的vertex事件依發生時的掃描線y排列進邊界的vertex事件佇列裡

//Input：邊界的vertex事件佇列(\*\*bequeue)和它的長度(\*index5)、要插入的邊界的vertex事件(\*bev)

//Output：NONE

insertbe(\*\* bequeue, \* bev, \* index5)

if (\*index5 == 0)

bequeue[0] = bev

(\*index5)++

return

if (bequeue[0]->y >= bev->y)

for (int j = \*index5; j > 0; j--)

bequeue[j] = bequeue[j - 1]

bequeue[0] = bev

(\*index5)++

return

if (bequeue[\*index5 - 1]->y <= bev->y)

bequeue[\*index5] = bev

(\*index5)++

return

for (int i = 0; i < \*index5 - 1; i++)

if (bequeue[i]->y <= bev->y && bequeue[i + 1]->y >= bev->y)

for (int j = \*index5; j > i + 1; j--)

bequeue[j] = bequeue[j - 1]

bequeue[i + 1] = bev

(\*index5)++

break

//判斷是否有vertex事件發生，並記錄它發生時的掃描線y及vertex點座標和產生它的三個拋物線

//Input：整條beachline(\* b)、當下的掃描線(y0)、vertex事件佇列(\*\*equeue)和它的長度(\*index3)

//Output：NONE

circle\_event(\* b, y0, \*\* equeue, \* index3)

if (!\*b || !(\*b)->lp || !(\*b)->rp)

return

if (circle((\*b)->lp->p, (\*b)->p, (\*b)->rp->p, &y, &o) && y >= y0)

ev = (e\*)malloc(sizeof(\*ev))

ev->valid = true

ev->y = y

ev->o = o

ev->b = \*b

(\*b)->ev = ev

inserte(equeue, ev, index3)

//判斷三拋物線是否有斷點，並產生發生時的掃描線y及vertex點座標

//Input：beachlinex比較小的拋物線的焦點(a) ，beachlinex中間的拋物線的焦點(b)，beachlinex比較大的拋物線的焦點(c)，用於紀錄產生的發生時的掃描線y(\*y)，用於紀錄產生的vertex點座標(\*o)

//Output：有斷點則true，沒斷點則false

circle(a, b, c, \* y, \* o)

if ((b.y - a.y) \* (c.x - a.x) - (c.y - a.y) \* (b.x - a.x) > 0)

return false

A = 2 \* (b.x - a.x)

B = 2 \* (b.y - a.y)

A2 = 2 \* (c.x - b.x)

B2 = 2 \* (c.y - b.y)

C = b.x \* b.x + b.y \* b.y - a.x \* a.x - a.y \* a.y

C2 = c.x \* c.x + c.y \* c.y - b.x \* b.x - b.y \* b.y

D = (A \* B2 - A2 \* B)

if (D == 0)

return false

o->x = (B2 \* C - B \* C2) / D

o->y = (A \* C2 - A2 \* C) / D

\*y = o->y + sqrt(pow(a.y - o->y, 2) + pow(a.x - o->x, 2))

return true

//將拋物線插入beachline的某拋物線後

//Input：要插入的拋物線的焦點(p)、p該插入的beachline的拋物線位置(\*b)

//Output：NONE

insertion(p, \* b)

if ((\*b)->bev)

(\*b)->bev->valid = false

(\*b)->bev = NULL

if ((\*b)->bevd)

(\*b)->bevd->valid = false

(\*b)->bevd = NULL

if ((\*b)->bevr)

(\*b)->bevr->valid = false

(\*b)->bevr = NULL

if ((\*b)->bevl)

(\*b)->bevl->valid = false

(\*b)->bevl = NULL

if ((\*b)->ev)

(\*b)->ev->valid = false

(\*b)->ev = NULL

if ((\*b)->rp && (\*b)->rp->ev)

(\*b)->rp->ev->valid = false

(\*b)->rp->ev = NULL

j = \*b

i = (bline)malloc(sizeof(\*i))

i->p = p

i->lp = i->rp = NULL

i->ev = NULL

i->bev = NULL

i->bevd = NULL

i->bevr = NULL

i->bevl = NULL

i->lp = j

if ((\*b)->rp)

i->rp = j->rp

j->rp->lp = i

j->rp = i

//將新拋物線插入beachline，並重新計算鄰近的拋物線的所有事件

//Input：該插入的點(p)，整條beachline(\*b)，vertex事件的佇列(\*\*ev)及它的長度(\*index3) ，邊界的vertex事件的佇列(\*\*bev)及它的長度(\*index5)

//Output：NONE

Insert( p, \* b, \*\* ev, \* index3, \*\* bev, \* index5)

if (!\*b)

bline a = NULL

a = (bline)malloc(sizeof(\*a))

a->p = p

a->lp = a->rp = NULL

a->ev = NULL

a->bev = NULL

a->bevd = NULL

a->bevr = NULL

a->bevl = NULL

\*b = a

return

iftwo = false

j = 0

for (i = \*b; i; i = i->rp, j++)

if (ifintersect(p, i))

if (j == 0)

if (intersection(p, i->p, p.y).y > 0)

a = (bline)malloc(sizeof(\*a))

a->p = i->p

a->lp = a->rp = NULL

a->ev = NULL

a->bev = NULL

a->bevd = NULL

a->bevr = NULL

a->bevl = NULL

a->rp = i

i->lp = a

insertion(p, &i->lp)

boundu\_event(&i->lp, bev, index5, p.y)

boundu\_event(&i->lp->lp, bev, index5, p.y)

boundr\_event(&i->lp, bev, index5, p.y)

boundr\_event(&i->lp->lp, bev, index5, p.y)

boundl\_event(&i->lp, bev, index5, p.y)

boundl\_event(&i->lp->lp, bev, index5, p.y)

boundd\_event(&i->lp, bev, index5, p.y)

boundd\_event(&i->lp->lp, bev, index5, p.y)

circle\_event(&i, p.y, ev, index3)

circle\_event(&i->lp, p.y, ev, index3)

\*b = i->lp->lp

else if (intersection(p, i->p, p.y).x > i->p.x)

insertion(p, &i)

boundu\_event(&i, bev, index5, p.y)

boundu\_event(&i->rp, bev, index5, p.y)

boundd\_event(&i, bev, index5, p.y)

boundd\_event(&i->rp, bev, index5, p.y)

boundr\_event(&i, bev, index5, p.y)

boundr\_event(&i->rp, bev, index5, p.y)

boundl\_event(&i, bev, index5, p.y)

boundl\_event(&i->rp, bev, index5, p.y)

circle\_event(&i->rp, p.y, ev, index3)

circle\_event(&i->rp->rp, p.y, ev, index3)

else

a = (bline)malloc(sizeof(\*a))

a->p = p

a->lp = a->rp = NULL

a->ev = NULL

a->bev = NULL

a->bevd = NULL

a->bevr = NULL

a->bevl = NULL

a->rp = i

i->lp = a

boundu\_event(&i->lp, bev, index5, p.y)

boundd\_event(&i->lp, bev, index5, p.y)

boundr\_event(&i->lp, bev, index5, p.y)

boundl\_event(&i->lp, bev, index5, p.y)

circle\_event(&i, p.y, ev, index3)

\*b = i->lp

return

if (!i->rp)

if (intersection(i->p, p, p.y).y > 0)

insertion(i->p, &i)

insertion(p, &i)

boundu\_event(&i, bev, index5, p.y)

boundu\_event(&i->rp, bev, index5, p.y)

boundd\_event(&i, bev, index5, p.y)

boundd\_event(&i->rp, bev, index5, p.y)

boundr\_event(&i, bev, index5, p.y)

boundr\_event(&i->rp, bev, index5, p.y)

boundl\_event(&i, bev, index5, p.y)

boundl\_event(&i->rp, bev, index5, p.y)

circle\_event(&i, p.y, ev, index3)

circle\_event(&i->rp, p.y, ev, index3)

else if (intersection(p, i->p, p.y).x > i->p.x)

insertion(p, &i)

boundu\_event(&i, bev, index5, p.y)

boundd\_event(&i, bev, index5, p.y)

boundr\_event(&i, bev, index5, p.y)

boundl\_event(&i, bev, index5, p.y)

circle\_event(&i, p.y, ev, index3)

else

insertion(p, &i->lp)

boundu\_event(&i->lp, bev, index5, p.y)

boundu\_event(&i->lp->lp, bev, index5, p.y)

boundr\_event(&i->lp, bev, index5, p.y)

boundr\_event(&i->lp->lp, bev, index5, p.y)

boundd\_event(&i->lp, bev, index5, p.y)

boundd\_event(&i->lp->lp, bev, index5, p.y)

boundl\_event(&i->lp, bev, index5, p.y)

boundl\_event(&i->lp->lp, bev, index5, p.y)

circle\_event(&i->lp, p.y, ev, index3)

circle\_event(&i->lp->lp, p.y, ev, index3)

return

if (i->rp && intersection(p, i->rp->p, p.y).y < intersection(p, i->p, p.y).y)

insertion(i->p, &i)

iftwo = true

insertion(p, &i)

boundu\_event(&i, bev, index5, p.y)

boundu\_event(&i->rp, bev, index5, p.y)

boundd\_event(&i, bev, index5, p.y)

boundd\_event(&i->rp, bev, index5, p.y)

boundr\_event(&i, bev, index5, p.y)

boundr\_event(&i->rp, bev, index5, p.y)

boundl\_event(&i, bev, index5, p.y)

boundl\_event(&i->rp, bev, index5, p.y)

circle\_event(&i->rp, p.y, ev, index3)

circle\_event(&i, p.y, ev, index3)

circle\_event(&i->rp->rp, p.y, ev, index3)

if (iftwo)

boundu\_event(&i->rp->rp, bev, index5, p.y)

boundd\_event(&i->rp->rp, bev, index5, p.y)

boundl\_event(&i->rp->rp, bev, index5, p.y)

boundr\_event(&i->rp->rp, bev, index5, p.y)

circle\_event(&i->rp->rp->rp, p.y, ev, index3)

return

insertion(p, b)

boundu\_event(b, bev, index5, p.y)

boundr\_event(b, bev, index5, p.y)

boundl\_event(b, bev, index5, p.y)

boundd\_event(b, bev, index5, p.y)

//判斷是否該插入在這裡

//Input：想插入的點(p1)、想插入的beachline的位置拋物線(p2)

//Output：該插入在這裡則return true，否則return false

ifintersect(p1, p2)

if (p2->p.y == p1.y)

return false

if (p2->lp)

a = intersection(p2->lp->p, p2->p, p1.y).x

if (p2->rp)

b = intersection(p2->p, p2->rp->p, p1.y).x

if ((!p2->lp || a <= p1.x) && (!p2->rp || p1.x <= b))

return true

return false

//計算兩拋物線的斷點

//beachlinex比較小的拋物線的焦點(p1)，beachlinex比較大的拋物線的焦點(p2)、當下的掃描線(d)

//Output：兩拋物線斷點(p)

intersection(p1, p2, d) {

if (p1.y == p2.y)

p.x = (p1.x + p2.x) / 2

else if (p2.y == d)

p.x = p2.x

else if (p1.y == d) {

p.x = p1.x

p1 = p2

else {

z0 = 2 \* (p1.y - d)

z1 = 2 \* (p2.y - d)

a = 1 / z0 - 1 / z1

b = -2 \* (p1.x / z0 - p2.x / z1)

c = (p1.x \* p1.x + p1.y \* p1.y - d \* d) / z0 - (p2.x \* p2.x + p2.y \* p2.y - d \* d) / z1

x1 = (-b - sqrt(b \* b - 4 \* a \* c)) / (2 \* a)

x2 = (-b + sqrt(b \* b - 4 \* a \* c)) / (2 \* a)

if (p1.x == p2.x)

if (p1.y < p2.y)

if (x1 < x2)

p.x = x1

else

p.x = x2

else

if (x1 < x2)

p.x = x2

else

p.x = x1

else if (p1.x > p2.x)

if (p1.y < p2.y)

if (x1 < x2)

p.x = x1

else

p.x = x2

else

if (x1 < x2)

p.x = x2

else

p.x = x1

else

if (p1.y < p2.y)

if (x1 < x2)

p.x = x1

else

p.x = x2

else

if (x1 < x2)

p.x = x2

else

p.x = x1

p.y = (p1.y \* p1.y + (p1.x - p.x) \* (p1.x - p.x) - d \* d) / (2 \* p1.y - 2 \* d)

return p

1. 演算法時間複雜度(time complexity)

找一遍地圖是O(a\*b)，做fortune algorithm是O(nlogn)。

所以是O(a\*b)+O(nlogn)。

1. 演算法空間複雜度(space complexity)

Beachline需O(n)的空間

vertex事件因為最多每相鄰三個有一個，所以是O(n)的空間

邊界的vertex事件因為最多每相鄰兩個有一個，所以是O(n)的空間

Vertex點需約O(n)的空間

邊界的Vertex點需約O(n)的空間

所以O(n)+O(n)+O(n)+O(n)+O(n)=O(n)

(2) 依序列出充電站的服務範圍(面積)。

\*因為程式碼太長，在(1)已有的程式碼就不放上來了\*

Input：一張a\*b具有n個充電站的地圖(單位公里)，x=0代表空的，x>0代表充電站。

Output：所有充電站的服務範圍。

Index0=0,index7=0

for i=0 to a do

for j=0 to b do

if(map[i][j]>0)

p[index].x=j

p[index++].y=i

p2[index7].x=j

p2[index7++].y=i

index2=0,index3=0,index4=0,index5=0,index6=0

while (index != 0)

if (index3 != 0 && etop(ev)->y <= ptop(p).y && etop(ev)->y <= betop(bev)->y)

process\_event(ev, &index3, vt, &index4, bev, &index5, bvt, &index6)

else if (index5 == 0 || ptop(p).y <= betop(bev)->y)

process\_point(p, &index, &beachline, ev, &index3, bev, &index5)

else if (index5 != 0)

process\_bevent(bev, &index5, bvt, &index6)

while (index3 != 0 || index5 != 0)

if (index5 == 0 || (index3 != 0 && etop(ev)->y <= betop(bev)->y))

process\_event(ev, &index3, vt, &index4, bev, &index5, bvt, &index6)

else if (index5 != 0)

process\_bevent(bev, &index5, bvt, &index6)

printf("點為：\n")

for i = 0 to index4-1 do

printf("(%.2f,%.2f)\n", vt[i].p.x, vt[i].p.y)

printf("邊界點為：\n")

for i = 0 to index6-1 do

printf("(%.2f,%.2f)\n", bvt[i].p.x, bvt[i].p.y)

print(bvt, index6, vt, index4)

area(p2, index7, vt, index4, bvt, index6)

freeb(&beachline)

//計算每個充電站的服務範圍，並印出

//Input：所有充電站點陣列(\* p)以及它的長度(index7)、所有vertex點陣列( \* vt)以及它的長度(index4)、所有邊界的vertex點陣列( \* bvt)以及它的長度( index6)

Output：NONE

area(\* p,index7, \* vt,index4, \* bvt, index6)

index = 0;

index2 = 0;

index3 = 0;

index5 = 0;

int index8 = 0;

min = maxx \* maxx + maxy \* maxy

for i = 0 to index7-1 do

if (p[i].x \* p[i].x + p[i].y \* p[i].y <= min)

min = p[i].x \* p[i].x + p[i].y \* p[i].y

for i = 0 to index7-1 do

if (p[i].x \* p[i].x + p[i].y \* p[i].y == min)

px0y0[index2++] = p[i]

min = maxx \* maxx + maxy \* maxy

for i = o to i < index7-1 do

if (p[i].x \* p[i].x + pow(p[i].y - maxy, 2) <= min)

min = p[i].x \* p[i].x + pow(p[i].y - maxy, 2)

for = 0 to index7-1 do

if (p[i].x \* p[i].x + pow(p[i].y - maxy, 2) == min)

px0ym[index3++] = p[i]

min = maxx \* maxx + maxy \* maxy

for i = 0 to index7-1 do

if (p[i].y \* p[i].y + pow(p[i].x - maxx, 2) <= min)

min = p[i].y \* p[i].y + pow(p[i].x - maxx, 2)

for i = 0 to index7-1 do

if (p[i].y \* p[i].y + pow(p[i].x - maxx, 2) == min)

pxmy0[index5++] = p[i]

min = maxx \* maxx + maxy \* maxy

for i = 0 to index7-1 do

if (pow(p[i].x - maxx, 2) + pow(p[i].y - maxy, 2) <= min)

min = pow(p[i].x - maxx, 2) + pow(p[i].y - maxy, 2)

for i = 0 to index7-1 do

if (pow(p[i].x - maxx, 2) + pow(p[i].y - maxy, 2) == min)

pxmym[index8++] = p[i]

for i = 0 to index7-1 do

for j = 0 to index4-1

if ((vt[j].p1.x == p[i].x && vt[j].p1.y == p[i].y) || (vt[j].p2.x == p[i].x && vt[j].p2.y == p[i].y) || (vt[j].p3.x == p[i].x && vt[j].p3.y == p[i].y))

pv[index++] = vt[j].p

for j = 0 to index6-1 do

if ((bvt[j].p1.x == p[i].x && bvt[j].p1.y == p[i].y) || (bvt[j].p2.x == p[i].x && bvt[j].p2.y == p[i].y))

pv[index++] = bvt[j].p

for j = 0 to index2-1 do

if (px0y0[j].x == p[i].x && px0y0[j].y == p[i].y)

pv[index].x = 0

pv[index++].y = 0

break

for j = 0 to index3-1 do

if (px0ym[j].x == p[i].x && px0ym[j].y == p[i].y)

pv[index].x = 0

pv[index++].y = maxy

break

for j = 0 to index5-1 do

if (pxmy0[j].x == p[i].x && pxmy0[j].y == p[i].y)

pv[index].x = maxx

pv[index++].y = 0

break

for j = 0 to index8-1 do

if (pxmym[j].x == p[i].x && pxmym[j].y == p[i].y)

pv[index].x = maxx

pv[index++].y = maxy

break

center.x = 0

center.y = 0

for j = 0 to index - 1 do

center.x += pv[j].x

center.y += pv[j].y

center.x /= index

center.y /= index

for j = 0 to index–2 do

for (int k = 0; k < index - j - 1; k++)

if ((pv[k].x - center.x) \* (pv[k + 1].y - center.y) - (pv[k + 1].x - center.x) \* (pv[k].y - center.y) < 0)

point temp

temp = pv[k]

pv[k] = pv[k + 1]

pv[k + 1] = temp

a = 0

for j = 0 to index-1 do

a += pv[j % index].x \* pv[(j + 1) % index].y - pv[(j + 1) % index].x \* pv[j % index].y

a /= 2

ar[index9++] = a;

index = 0

for i = 0 to index9 – 2 do

for j = 0 to index9 - i – 2 do

if (ar[j + 1] < ar[j])

emp = ar[j]

t = p[j]

p[j] = p[j + 1]

p[j + 1] = t

ar[j] = ar[j + 1]

ar[j + 1] = temp

for i = 0 to index7-1 do

printf("充電站(%.0f,%.0f)的服務範圍為%f平方公里\n", p[i].x, p[i].y, ar[i])

1. 演算法時間複雜度(time complexity)

找一遍地圖是O(a\*b)，做fortune是O(nlogn)，排序為逆時針是O(n3)，計算面積是O(n2)，排序面積是O(n2)。

所以O(a\*b)+O(nlogn)+ O(n3)+ O(n2)+ O(n2)= O(a\*b)+O(n3)。

1. 演算法空間複雜度(space complexity)

Beachline需O(n)的空間

vertex事件因為最多每相鄰三個有一個，所以是O(n)的空間

邊界的vertex事件因為最多每相鄰兩個有一個，所以是O(n)的空間

Vertex點需約O(n)的空間

邊界的Vertex點需約O(n)的空間

計算面積約需O(n)的空間

所以O(n)+O(n)+O(n)+O(n)+O(n) +O(n)=O(n)

(3) 找出離充電站最遠的點以及距離(Largest Empty Circle Problem with Voronoi Diagram, also called toxic waste dump problem)。

\*因為程式碼太長，在(1)(2)已有的程式碼就不放上來了\*

Input：一張a\*b具有n個充電站的地圖(單位公里)，x=0代表空的，x>0代表充電站。

Output：離充電站最遠的點以及距離

Index0=0,index7=0

for i=0 to a do

for j=0 to b do

if(map[i][j]>0)

p[index].x=j

p[index++].y=i

p2[index7].x=j

p2[index7++].y=i

index2=0,index3=0,index4=0,index5=0,index6=0

while (index != 0)

if (index3 != 0 && etop(ev)->y <= ptop(p).y && etop(ev)->y <= betop(bev)->y)

process\_event(ev, &index3, vt, &index4, bev, &index5, bvt, &index6)

else if (index5 == 0 || ptop(p).y <= betop(bev)->y)

process\_point(p, &index, &beachline, ev, &index3, bev, &index5)

else if (index5 != 0)

process\_bevent(bev, &index5, bvt, &index6)

while (index3 != 0 || index5 != 0)

if (index5 == 0 || (index3 != 0 && etop(ev)->y <= betop(bev)->y))

process\_event(ev, &index3, vt, &index4, bev, &index5, bvt, &index6)

else if (index5 != 0)

process\_bevent(bev, &index5, bvt, &index6)

printf("點為：\n")

for i = 0 to index4-1 do

printf("(%.2f,%.2f)\n", vt[i].p.x, vt[i].p.y)

printf("邊界點為：\n")

for i = 0 to index6-1 do

printf("(%.2f,%.2f)\n", bvt[i].p.x, bvt[i].p.y)

print(bvt, index6, vt, index4)

dt(vt, index4, bvt, index6, p2, index7)

freeb(&beachline)

//找到離充電站最遠的點，並印出此點座標與它和充電站的距離

//Input：vertex點的陣列(\*vt)及它的長度(index4)、邊界的vertex點的陣列(\*bvt)及它的長度(index6)、充電站點的陣列(\*p)以及它的長度(index)

//Output：NONE

dt(\* vt, index4, \* bvt, index6, \* p, index) {

max = 0

for i = 0 to index4-1 do

if (max <= pow(vt[i].p1.x - vt[i].p.x, 2) + pow(vt[i].p1.y - vt[i].p.y, 2))

ife2 = true

for j = 0 to index-1 do

if (vt[i].p.x == p[j].x && vt[i].p.y == p[j].y)

ife2 = false

break

if (ife2)

max = pow(vt[i].p1.x - vt[i].p.x, 2) + pow(vt[i].p1.y - vt[i].p.y, 2)

o = vt[i].p

for i = 0 to index6-1 do

if (max <= pow(bvt[i].p1.x - bvt[i].p.x, 2) + pow(bvt[i].p1.y - bvt[i].p.y, 2))

bool ife2 = true

for j = 0 to index-1 do

if (bvt[i].p.x == p[j].x && bvt[i].p.y == p[j].y)

ife2 = false

break

if (ife2)

max = pow(bvt[i].p1.x - bvt[i].p.x, 2) + pow(bvt[i].p1.y - bvt[i].p.y, 2)

o = bvt[i].p

min = maxx \* maxx + maxy \* maxy

ife = true

for i = 0 to index-1 do

if (p[i].x == 0 && p[i].y == 0)

ife = false

break

if (ife)

for i = 0 to index-1 do

if (min >= pow(p[i].x, 2) + pow(p[i].y, 2))

min = pow(p[i].x, 2) + pow(p[i].y, 2)

if (max < min)

max = min

o.x = 0

o.y = 0

ife = true

min = maxx \* maxx + maxy \* maxy

for i = 0 to index-1 do

if (p[i].x == 0 && p[i].y == maxy)

ife = false

break

if (ife)

for i = 0 to index-1 do

if (min >= pow(p[i].x, 2) + pow(p[i].y - maxy, 2))

min = pow(p[i].x, 2) + pow(p[i].y - maxy, 2)

if (max < min)

max = min

o.x = 0

o.y = maxy

ife = true

min = maxx \* maxx + maxy \* maxy

for i = 0 to index-1 do

if (p[i].x == maxx && p[i].y == maxy)

ife = false

break

if (ife)

for i = 0 to index-1 do

if (min >= pow(p[i].x - maxx, 2) + pow(p[i].y - maxy, 2))

min = pow(p[i].x - maxx, 2) + pow(p[i].y - maxy, 2)

if (max <= min)

max = min

o.x = maxx

o.y = maxy

ife = true

min = maxx \* maxx + maxy \* maxy

for (int i = 0; i < index; i++)

if (p[i].x == maxx && p[i].y == 0)

ife = false

break

if (ife)

for i = 0 to index-1 do

if (min >= pow(p[i].x - maxx, 2) + pow(p[i].y, 2))

min = pow(p[i].x - maxx, 2) + pow(p[i].y, 2)

if (max < min)

max = min

o.x = maxx

o.y = 0

printf("離充電站最遠的點為(%.2f,%.2f)距離%f公里\n", o.x, o.y, sqrt(max))

1. 演算法時間複雜度(time complexity)

找一遍地圖是O(a\*b)，做fortune是O(nlogn)，計算vertex是O(n)，計算四個角是O(n)。

所以是O(a\*b)+O(nlogn)+O(n)+O(n)= O(a\*b)+O(nlogn)。

1. 演算法空間複雜度(space complexity)

Beachline需O(n)的空間

vertex事件因為最多每相鄰三個有一個，所以是O(n)的空間

邊界的vertex事件因為最多每相鄰兩個有一個，所以是O(n)的空間

Vertex點需約O(n)的空間

邊界的Vertex點需約O(n)的空間

找離充電站最遠的vertex需約O(1)的空間

所以O(n)+O(n)+O(n)+O(n)+O(n)+O(1)=O(n)

(4) 根據Sub-problem (3)，在該位置安裝一個充電站，則須在原來的Voronoi Diagram外，加一個點變成有*n+1*個點的Voronoi Diagram，設計一程式法建構該新的Voronoi Diagram。

Input：一張a\*b具有n個充電站的地圖(單位公里)，x=0代表空的，x>0代表充電站。

Output：加入離充電站最遠的點後的vertex點和邊界的vertex點

\*因為程式碼太長，在(1)(2)(3)已有的程式碼就不放上來了\*

Index0=0,index7=0

for i=0 to a do

for j=0 to b do

if(map[i][j]>0)

p[index].x=j

p[index++].y=i

p2[index7].x=j

p2[index7++].y=i

index2=0,index3=0,index4=0,index5=0,index6=0

while (index != 0)

if (index3 != 0 && etop(ev)->y <= ptop(p).y && etop(ev)->y <= betop(bev)->y)

process\_event(ev, &index3, vt, &index4, bev, &index5, bvt, &index6)

else if (index5 == 0 || ptop(p).y <= betop(bev)->y)

process\_point(p, &index, &beachline, ev, &index3, bev, &index5)

else if (index5 != 0)

process\_bevent(bev, &index5, bvt, &index6)

while (index3 != 0 || index5 != 0)

if (index5 == 0 || (index3 != 0 && etop(ev)->y <= betop(bev)->y))

process\_event(ev, &index3, vt, &index4, bev, &index5, bvt, &index6)

else if (index5 != 0)

process\_bevent(bev, &index5, bvt, &index6)

printf("點為：\n")

for i = 0 to index4-1 do

printf("(%.2f,%.2f)\n", vt[i].p.x, vt[i].p.y)

printf("邊界點為：\n")

for i = 0 to index6-1 do

printf("(%.2f,%.2f)\n", bvt[i].p.x, bvt[i].p.y)

freeb(&beachline)

print(bvt, index6, vt, index4)

index8 = 0

op = dt(vt, &index4, bvt, &index6, p2, index7, p3, &index8)

printf("新增點(%.2f,%.2f)\n", op.x, op.y)

insertv(op, p2, &index7)

for i = 0 to index4-1 do

if (pow(vt[i].p.x - vt[i].p1.x, 2) + pow(vt[i].p.y - vt[i].p1.y, 2) > pow(op.x - vt[i].p.x, 2) + pow(op.y - vt[i].p.y, 2) || pow(vt[i].p.x - vt[i].p2.x, 2) + pow(vt[i].p.y - vt[i].p2.y, 2) > pow(op.x - vt[i].p.x, 2) + pow(op.y - vt[i].p.y, 2) || pow(vt[i].p.x - vt[i].p3.x, 2) + pow(vt[i].p.y - vt[i].p3.y, 2) > pow(op.x - vt[i].p.x, 2) + pow(op.y - vt[i].p.y, 2))

insertv(vt[i].p1, p3, &index8)

insertv(vt[i].p2, p3, &index8)

insertv(vt[i].p3, p3, &index8)

dv(vt, i, &index4)

i--

for i = 0 to index6-1 do

if (pow(bvt[i].p.x - bvt[i].p1.x, 2) + pow(bvt[i].p.y - bvt[i].p1.y, 2) > pow(op.x - bvt[i].p.x, 2) + pow(op.y - bvt[i].p.y, 2) || pow(bvt[i].p.x - bvt[i].p2.x, 2) + pow(bvt[i].p.y - bvt[i].p2.y, 2) > pow(op.x - bvt[i].p.x, 2) + pow(op.y - bvt[i].p.y, 2))

insertv(bvt[i].p1, p3, &index8)

insertv(bvt[i].p2, p3, &index8)

dbv(bvt, i, &index6)

i--;

b2 = NULL

index9 = index6

index10=index4

while (index8 != 0)

if (index3 != 0 && etop(ev)->y <= ptop(p3).y && etop(ev)->y <= betop(bev)->y)

process\_event(ev, &index3, vt, &index4, bev, &index5, bvt, &index6)

else if (index5 == 0 || ptop(p3).y <= betop(bev)->y)

process\_point(p3, &index8, &b2, ev, &index3, bev, &index5)

else if (index5 != 0)

process\_bevent(bev, &index5, bvt, &index6)

while (index3 != 0 || index5 != 0)

if (index5 == 0 || (index3 != 0 && etop(ev)->y <= betop(bev)->y))

process\_event(ev, &index3, vt, &index4, bev, &index5, bvt, &index6)

else if (index5 != 0)

process\_bevent(bev, &index5, bvt, &index6)

checkdbv(p2, bvt, index9, &index6, index7)

checkdv(p2, vt, index10, &index4, index7)

freeb(&b2)

printf("充電站位置：\n")

for i = 0 to index7-1 do

printf("(%.2f,%.2f)\n", p2[i].x, p2[i].y)

printf("點為：\n")

for i = 0 to index4-1 do

printf("(%.2f,%.2f)\n", vt[i].p.x, vt[i].p.y)

printf("邊界點為：\n")

for i = 0 to index6-1 do

printf("(%.2f,%.2f)\n", bvt[i].p.x, bvt[i].p.y)

print(bvt, index6, vt, index4)

//找到離充電站距離最遠的點，並將此點和產生它的拋物線焦點放到要討論的充電站點陣列中

//Input：vertex點的陣列(\*vt)及它的長度(\*index4)、邊界的vertex點的陣列(\*bvt)及它的長度(\*index6)、充電站點的陣列(\*p)以及它的長度(index)、要討論的充電站點陣列(\*p3)以及它的長度(\*index8)

//Output：離充電站距離最遠的點

dt(\* vt, \*index4, \* bvt, \*index6, \* p, index, \* p3, \*index8) {

max = 0;

for i = 0 to \*index4-1 do

if (max <= pow(vt[i].p1.x - vt[i].p.x, 2) + pow(vt[i].p1.y - vt[i].p.y, 2))

ife2 = true

for j = 0 to index-1 do

if (vt[i].p.x == p[j].x && vt[i].p.y == p[j].y)

ife2 = false

break

if (ife2)

max = pow(vt[i].p1.x - vt[i].p.x, 2) + pow(vt[i].p1.y - vt[i].p.y, 2)

o = vt[i].p

oindex = i

for i = 0 to \*index6-1 do

if (max <= pow(bvt[i].p1.x - bvt[i].p.x, 2) + pow(bvt[i].p1.y - bvt[i].p.y, 2))

bool ife2 = true;

for j = 0 to index-1 do

if (bvt[i].p.x == p[j].x && bvt[i].p.y == p[j].y)

ife2 = false

break

if (ife2)

max = pow(bvt[i].p1.x - bvt[i].p.x, 2) + pow(bvt[i].p1.y - bvt[i].p.y, 2)

o = bvt[i].p

oindex = i

w = 1

min = maxx \* maxx + maxy \* maxy

ife = true

for i = 0 to index-1 do

if (p[i].x == 0 && p[i].y == 0)

ife = false

break

if (ife)

for i = 0 to index-1 do

if (min >= pow(p[i].x, 2) + pow(p[i].y, 2))

min = pow(p[i].x, 2) + pow(p[i].y, 2)

if (max < min)

max = min

o.x = 0

o.y = 0

w = 2

ife = true

min = maxx \* maxx + maxy \* maxy

for i = 0 to index-1 do

if (p[i].x == 0 && p[i].y == maxy)

ife = false

break

if (ife)

for i = 0 to index-1 do

if (min >= pow(p[i].x, 2) + pow(p[i].y - maxy, 2))

min = pow(p[i].x, 2) + pow(p[i].y - maxy, 2)

if (max < min)

max = min

o.x = 0

o.y = maxy

w = 3

ife = true

min = maxx \* maxx + maxy \* maxy

for i = 0 to index-1 do

if (p[i].x == maxx && p[i].y == maxy)

ife = false

break

if (ife)

for i = 0 to index-1 do

if (min >= pow(p[i].x - maxx, 2) + pow(p[i].y - maxy, 2))

min = pow(p[i].x - maxx, 2) + pow(p[i].y - maxy, 2)

if (max <= min)

max = min

o.x = maxx

o.y = maxy

w = 4

ife = true;

min = maxx \* maxx + maxy \* maxy

for i = 0 to index-1 do

if (p[i].x == maxx && p[i].y == 0)

ife = false

break

if (ife)

for i = 0 to index-1 do

if (min >= pow(p[i].x - maxx, 2) + pow(p[i].y, 2))

min = pow(p[i].x - maxx, 2) + pow(p[i].y, 2)

if (max < min)

max = min

o.x = maxx

o.y = 0

w = 5

insertv(o, p3, index8)

switch (w)

case 0

insertv(vt[oindex].p1, p3, index8)

insertv(vt[oindex].p2, p3, index8)

insertv(vt[oindex].p3, p3, index8)

dv(vt, oindex, index4)

break

case 1:

insertv(bvt[oindex].p1, p3, index8)

insertv(bvt[oindex].p2, p3, index8)

dbv(bvt, oindex, index6)

break

case 2:

for i = 0 to index-1 do

if (max == pow(p[i].x, 2) + pow(p[i].y, 2))

insertv(p[i], p3, index8)

break

case 3:

for i = 0 to index-1 do

if (max == pow(p[i].x, 2) + pow(p[i].y - maxy, 2))

insertv(p[i], p3, index8)

break

case 4:

for i = 0 to index-1 do

if (max == pow(p[i].x - maxx, 2) + pow(p[i].y - maxy, 2))

insertv(p[i], p3, index8)

break

case 5:

for i = 0 to index-1 do

if (max == pow(p[i].x - maxx, 2) + pow(p[i].y, 2))

insertv(p[i], p3, index8)

break

return o

//把不該產生的邊界的vertex刪掉

//Input：充電站點陣列(\*p)以及它的長度(index7)、邊界的vertex點的陣列(\*bvt)及它的長度(\*index6)，舊的邊界的vertex的長度(index9)

//Output：NONE

checkdbv(\* p, \* bvt, index9, \* index6, index7)

for i = index9 to \*index6-1 do

for j = 0 to index9-1 do

if (bvt[i].p.x == bvt[j].p.x && bvt[i].p.y == bvt[j].p.y)

dbv(bvt, i--, index6)

break

for i = index9 to \*index6-1 do

min = pow(bvt[i].p1.x - bvt[i].p.x, 2) + pow(bvt[i].p1.y - bvt[i].p.y, 2);

for j = 0 to index7-1 do

if (bvt[i].p2.x != p[j].x || bvt[i].p2.y != p[j].y)

if (min > pow(p[j].x - bvt[i].p.x, 2) + pow(p[j].y - bvt[i].p.y, 2))

dbv(bvt, i--, index6)

break

//把不該產生的vertex刪掉

//Input：充電站點陣列(\*p)以及它的長度(index7)、vertex點的陣列(\*vt)及它的長度(\*index4)，舊的vertex的長度(index15)

//Output：NONE

checkdv(\* p, \* vt, index15, \* index4, index7)

for i = index15 to \*index4-1 do

for j = 0 to index15-1 do

if (vt[i].p.x == vt[j].p.x && vt[i].p.y == vt[j].p.y)

dv(vt, i--, index4)

break

for i = index15 to \*index4-1 do

min = pow(vt[i].p1.x - vt[i].p.x, 2) + pow(vt[i].p1.y - vt[i].p.y, 2)

for j = 0 to index7-1 do

if ((vt[i].p2.x != p[j].x || vt[i].p2.y != p[j].y) && (vt[i].p3.x != p[j].x || vt[i].p3.y != p[j].y))

if (min > pow(p[j].x - vt[i].p.x, 2) + pow(p[j].y - vt[i].p.y, 2))

dv(vt, i--, index4)

break

//刪掉某個vertex

//Input：vertex點的陣列(\*vt)及它的長度(\*index4)、要刪掉的vertex的索引值(i)

//Output：NONE

dv(\* vt, i, \* index4)

for j = i to \*index4 – 2 do

vt[j] = vt[j + 1]

(\*index4)—

//刪掉某個邊界的vertex

//Input：邊界的vertex點的陣列(\*bvt)及它的長度(\*index6)、要刪掉的邊界的vertex的索引值(i)

//Output：NONE

dbv(\* bvt, i, \* index6)

for j = i to \*index6 – 2 do

bvt[j] = bvt[j + 1]

(\*index6)—

//把點加入要討論的充電站點的陣列

//Input：要加入的點(p)、要討論的充電站點的陣列(\*p2)以及它的長度(\*index8)

//Output：NONE

insertv(p, \* p2, \* index8)

if (\*index8 == 0)

p2[(\*index8)++] = p

return

bool ife = true

for i = 0 to \*index8-1 do

if (p2[i].x == p.x && p2[i].y == p.y)

ife = false

break

if (ife)

if (p2[\*index8 - 1].y < p.y || (p2[\*index8 - 1].y == p.y && p2[\*index8 - 1].x <= p.x))

p2[(\*index8)++] = p

else if (p2[0].y > p.y || (p2[0].x >= p.x && p2[0].y == p.y))

for i = \*index8 downto 1 do

p2[i] = p2[i - 1]

p2[0] = p

(\*index8)++

else

for i = 0 to \*index8 – 2 do

if (p2[i].y < p.y && p2[i + 1].y > p.y)

x = i + 1

break

else if (p2[i].y == p.y && p2[i].x <= p.x && p2[i + 1].y == p.y && p2[i + 1].x >= p.x)

x = i + 1

break

else if (p2[i].y == p.y && p2[i].x <= p.x && p2[i + 1].y > p.y)

x = i + 1

break

else if (p2[i].y < p.y&& p2[i + 1].x >= p.x && p2[i + 1].y == p.y)

x = i + 1

break

for i = \*index8 downto x+1 do

p2[i] = p2[i - 1]

p2[x] = p

(\*index8)++

1. 演算法時間複雜度(time complexity)

找一遍地圖是O(a\*b)，做fortune是O(nlogn)，計算vertex是O(n)，計算四個角是O(n)，判斷是否有vertex與產生它的拋物線的焦點的距離大於這些vertex與新點的距離是O(n)。

所以O(a\*b)+O(nlogn)+O(n)+O(n)+O(n)= O(a\*b)+O(nlogn)。

1. 演算法空間複雜度(space complexity)

Beachline需O(n)的空間

vertex事件因為最多每相鄰三個有一個，所以是O(n)的空間

邊界的vertex事件因為最多每相鄰兩個有一個，所以是O(n)的空間

Vertex點需約O(n)的空間

邊界的Vertex點需約O(n)的空間

新增離充電站最遠的vertex需約O(n)的空間

所以O(n)+O(n)+O(n)+O(n)+O(n)+O(n)=O(n)

(5) 重複Sub-problems (3) and (4)直到最遠距離滿足一個門檻值(Threshold value *k*, 如500m，*k*值可自行設定)，並輸出總共需要幾個充電站、目前離充電站最遠的距離與*k*值。

Input：一張a\*b具有n個充電站的地圖(單位公里)，x=0代表空的，x>0代表充電站。

Output：重複加入離充電站最遠的點直到離充電站最遠的點的距離小於等於k(k0值)的vertex點和邊界的vertex點

\*因為程式碼太長，在(1)(2)(3)(4)已有的程式碼就不放上來了\*

index0=0,index7=0

for i=0 to a do

for j=0 to b do

if(map[i][j]>0)

p[index].x=j

p[index++].y=i

p2[index7].x=j

p2[index7++].y=i

index2=0,index3=0,index4=0,index5=0,index6=0

while (index != 0)

if (index3 != 0 && etop(ev)->y <= ptop(p).y && etop(ev)->y <= betop(bev)->y)

process\_event(ev, &index3, vt, &index4, bev, &index5, bvt, &index6)

else if (index5 == 0 || ptop(p).y <= betop(bev)->y)

process\_point(p, &index, &beachline, ev, &index3, bev, &index5)

else if (index5 != 0)

process\_bevent(bev, &index5, bvt, &index6)

while (index3 != 0 || index5 != 0)

if (index5 == 0 || (index3 != 0 && etop(ev)->y <= betop(bev)->y))

process\_event(ev, &index3, vt, &index4, bev, &index5, bvt, &index6)

else if (index5 != 0)

process\_bevent(bev, &index5, bvt, &index6)

printf("點為：\n")

for i = 0 to index4-1 do

printf("(%.2f,%.2f)\n", vt[i].p.x, vt[i].p.y)

printf("邊界點為：\n")

for i = 0 to index6-1 do

printf("(%.2f,%.2f)\n", bvt[i].p.x, bvt[i].p.y)

freeb(&beachline)

print(bvt, index6, vt, index4)

for i = 0 to index7-1 do

p4[index10++] = p2[i]

k = dtlong(vt, index4, bvt, index6, p4, index10)

index11 = 0, index12 = 0, index13 = 0, index14 = 0

for i = 0 to index4-1 do

vt2[index11++] = vt[i]

for i = 0 to index6-1 do

bvt2[index12++] = bvt[i]

while (k > k0)

index8 = 0;

op = dt(vt2, &index11, bvt2, &index12, p4, index10, p3, &index8)

printf("新增點(%.2f,%.2f)\n", op.x, op.y)

insertv(op, p4, &index10)

for i = 0 to index11-1 do

if (pow(vt2[i].p.x - vt2[i].p1.x, 2) + pow(vt2[i].p.y - vt2[i].p1.y, 2) > pow(op.x - vt2[i].p.x, 2) + pow(op.y - vt2[i].p.y, 2) || pow(vt2[i].p.x - vt2[i].p2.x, 2) + pow(vt2[i].p.y - vt2[i].p2.y, 2) > pow(op.x - vt2[i].p.x, 2) + pow(op.y - vt2[i].p.y, 2) || pow(vt2[i].p.x - vt2[i].p3.x, 2) + pow(vt2[i].p.y - vt2[i].p3.y, 2) > pow(op.x - vt2[i].p.x, 2) + pow(op.y - vt2[i].p.y, 2))

insertv(vt2[i].p1, p3, &index8)

insertv(vt2[i].p2, p3, &index8)

insertv(vt2[i].p3, p3, &index8)

dv(vt2, i, &index11)

i--

for i = 0 to index12-1 do

if (pow(bvt2[i].p.x - bvt2[i].p1.x, 2) + pow(bvt2[i].p.y - bvt2[i].p1.y, 2) > pow(op.x - bvt2[i].p.x, 2) + pow(op.y - bvt2[i].p.y, 2) || pow(bvt2[i].p.x - bvt2[i].p2.x, 2) + pow(bvt2[i].p.y - bvt2[i].p2.y, 2) > pow(op.x - bvt2[i].p.x, 2) + pow(op.y - bvt2[i].p.y, 2))

insertv(bvt2[i].p1, p3, &index8)

insertv(bvt2[i].p2, p3, &index8)

dbv(bvt2, i, &index12)

i--

b2 = NULL

index9 = index12

index15=index11

while (index8 != 0)

if (index13 != 0 && etop(ev2)->y <= ptop(p3).y && etop(ev2)->y <= betop(bev2)->y)

process\_event(ev2, &index13, vt2, &index11, bev2, &index14, bvt2, &index12)

else if (index14 == 0 || ptop(p3).y <= betop(bev2)->y)

process\_point(p3, &index8, &b2, ev2, &index13, bev2, &index14)

else if (index14 != 0)

process\_bevent(bev2, &index14, bvt2, &index12)

while (index13 != 0 || index14 != 0)

if (index14 == 0 || (index13 != 0 && etop(ev2)->y <= betop(bev2)->y))

process\_event(ev2, &index13, vt2, &index11, bev2, &index14, bvt2, &index12)

else if (index14 != 0)

process\_bevent(bev2, &index14, bvt2, &index12)

checkdbv(p4, bvt2, index9, &index12, index10)

checkdv(p4, vt2, index15, &index11, index10)

freeb(&b2)

n++

free(p3)

k = dtlong(vt2, index11, bvt2, index12, p4, index10)

free(ev2)

free(bev2)

printf("充電站位置：\n")

for i = 0 to index10-1 do

printf("(%.2f,%.2f)\n", p4[i].x, p4[i].y)

printf("總共需新增%d個充電站\n離充電站最遠的距離為%f公里\n點為：\n", n, k)

for i = 0 to index11-1 do

printf("(%.2f,%.2f)\n", vt2[i].p.x, vt2[i].p.y)

printf("邊界點為：\n")

for i = 0 to index12-1 do

printf("(%.2f,%.2f)\n", bvt2[i].p.x, bvt2[i].p.y)

print(bvt2, index12, vt2, index11)

//計算離充電站最遠的點的距離

//Input： vertex點的陣列(\*vt)及它的長度(index4)、邊界的vertex點的陣列(\*bvt)及它的長度(index6)、充電站點的陣列(\*p)以及它的長度(index)

//Output：離充電站最遠的點的距離

dtlong(\* vt, index4, \* bvt, index6, \* p, index)

max = 0

for i = 0 to index4-1 do

if (max <= pow(vt[i].p1.x - vt[i].p.x, 2) + pow(vt[i].p1.y - vt[i].p.y, 2))

ife2 = true

for j = 0 to index-1 do

if (vt[i].p.x == p[j].x && vt[i].p.y == p[j].y)

ife2 = false

break

if (ife2)

max = pow(vt[i].p1.x - vt[i].p.x, 2) + pow(vt[i].p1.y - vt[i].p.y, 2)

for i = 0 to index6-1 do

if (max <= pow(bvt[i].p1.x - bvt[i].p.x, 2) + pow(bvt[i].p1.y - bvt[i].p.y, 2))

ife2 = true

for j = 0 to index-1 do

if (bvt[i].p.x == p[j].x && bvt[i].p.y == p[j].y)

ife2 = false

break

if (ife2)

max = pow(bvt[i].p1.x - bvt[i].p.x, 2) + pow(bvt[i].p1.y - bvt[i].p.y, 2)

min = maxx \* maxx + maxy \* maxy

ife = true

for i = 0 to index-1 do

if (p[i].x == 0 && p[i].y == 0)

ife = false

break

if (ife)

for i = 0 to index-1 do

if (min >= pow(p[i].x, 2) + pow(p[i].y, 2))

min = pow(p[i].x, 2) + pow(p[i].y, 2)

if (max < min)

max = min

ife = true

min = maxx \* maxx + maxy \* maxy

for i = 0 to index-1 do

if (p[i].x == 0 && p[i].y == maxy)

ife = false

break

if (ife)

for i = 0 to index-1 do

if (min >= pow(p[i].x, 2) + pow(p[i].y - maxy, 2))

min = pow(p[i].x, 2) + pow(p[i].y - maxy, 2)

if (max < min)

max = min

ife = true

min = maxx \* maxx + maxy \* maxy

for i = 0 to index-1 do

if (p[i].x == maxx && p[i].y == maxy)

ife = false

break

if (ife)

for i = 0 to index-1 do

if (min >= pow(p[i].x - maxx, 2) + pow(p[i].y - maxy, 2))

min = pow(p[i].x - maxx, 2) + pow(p[i].y - maxy, 2)

if (max <= min)

max = min

ife = true

min = maxx \* maxx + maxy \* maxy

for i = 0 to index-1 do

if (p[i].x == maxx && p[i].y == 0)

ife = false

break

if (ife)

for i = 0 to index-1 do

if (min >= pow(p[i].x - maxx, 2) + pow(p[i].y, 2))

min = pow(p[i].x - maxx, 2) + pow(p[i].y, 2)

if (max < min)

max = min

return sqrt(max)

1. 演算法時間複雜度(time complexity)

找一遍地圖是O(a\*b)，做fortune是O(nlogn)，最多會重複O(n2/k2)次。

所以是O(a\*b)+O(n3logn/k2)。

1. 演算法空間複雜度(space complexity)

Beachline需O(n)的空間

vertex事件因為最多每相鄰三個有一個，所以是O(n)的空間

邊界的vertex事件因為最多每相鄰兩個有一個，所以是O(n)的空間

Vertex點需約O(n)的空間

邊界的Vertex點需約O(n)的空間

新增離充電站最遠的vertex最多需約O((a/k+1)\*(b/k+1))的空間

所以O(n)+O(n)+O(n)+O(n)+O(n)+O((a/k+1)\*(b/k+1))=O(n)+O((a\*b/k2))

**(四) 程式設計環境架構**

程式設計語言、工具、環境與電腦硬體等規格說明…

1. 程式語言

C in Windows 10

2. 程式開發工具

Visual Studio 2022

3. 電腦硬體

處理器 Intel(R) Core(TM) i5-9300H CPU @ 2.40GHz 2.40 GHz

已安裝記憶體(RAM) 8.00 GB (7.81 GB 可用)

系統類型 64 位元作業系統，x64 型處理器

手寫筆與觸控 手寫筆支援

(五) 程式 (含source code, input code, and output code)

程式含source code, input code, and output code等…

**(1) 依據充電站的位置(2D)，建構一個Voronoi Diagram with time complexity O(𝑛log𝑛).**

1. 主程式

C program

#define \_CRT\_SECURE\_NO\_WARNINGS

#include "math.h"

#include "stdio.h"

#include "stdlib.h"

#include "time.h"

int maxx = 0;

int maxy = 0;

struct point {

double x;

double y;

};

struct e;

struct be;

typedef struct bl\* bline;

typedef struct bl {

bline lp, rp;

point p;

e\* ev;

be\* bev;

be\* bevl;

be\* bevr;

be\* bevd;

};

struct e {

double y;

bline b;

point o;

bool valid = true;

};

struct be {

double y;

bline b1, b2;

point o;

bool valid;

};

struct v {

point p;

point p1, p2, p3;

};

struct bv {

point p;

point p1, p2;

};

point ppop(point\* p, int\* index);

point ptop(point\* p);

e\* epop(e\*\* ev, int\* index);

e\* etop(e\*\* ev);

void d(bline b);

void process\_point(point\* p, int\* index, bline\* b, e\*\* ev, int\* index3, be\*\* bev, int\* index5);

void process\_event(e\*\* equeue, int\* index3, v\* v, int\* index4, be\*\* bev, int\* index5, bv\* bvt, int\* index6);

void circle\_event(bline\* b, double y0, e\*\* equeue, int\* index3);

bool circle(point a, point b, point c, double\* y, point\* o);

void insertion(point p, bline\* b);

void insert(point p, bline\* b, e\*\* equeue, int\* index3, be\*\* bequeue, int\* index5);

bool ifintersect(point p1, bline p2);

point intersection(point p1, point p2, double d);

void insertbe(be\*\* bequeue, be\* bev, int\* index5);

be\* bepop(be\*\* bev, int\* index);

be\* betop(be\*\* bev);

bool boundl(point p1, point p2, double\* y, point\* o);

bool boundr(point p1, point p2, double\* y, point\* o);

bool boundu(point p1, point p2, double\* y, point\* o);

bool boundd(point p1, point p2, double\* y, point\* o);

void boundu\_event(bline\* b, be\*\* bequeue, int\* index5, double y0);

void process\_bevent(be\*\* bequeue, int\* index5, bv\* bv, int\* index6);

void boundu\_event(bline\* b, be\*\* bequeue, int\* index5, double y0);

void boundd\_event(bline\* b, be\*\* bequeue, int\* index5, double y0);

void boundr\_event(bline\* b, be\*\* bequeue, int\* index5, double y0);

void boundl\_event(bline\* b, be\*\* bequeue, int\* index5, double y0);

bool boundl(point p1, point p2, double\* y, point\* o);

bool boundr(point p1, point p2, double\* y, point\* o);

bool boundd(point p1, point p2, double\* y, point\* o);

bool boundu(point p1, point p2, double\* y, point\* o);

void inserte(e\*\* equeue, e\* ev, int\* index3);

void print(bv\* bvt, int index6, v\* vt, int index4);

void freeb(bline\* b);

int main(void) {

FILE\* map;

char s[10];

printf("請輸入檔案名稱(9位元內)>>");

scanf("%s", s);

map = fopen(s, "r");

int in;

char c;

int index = 0;

int ifeof = 1;

while (ifeof != EOF) {

ifeof = fscanf(map, "%d ", &in);

maxx++;

}

maxx--;

fseek(map, 0, SEEK\_SET);

ifeof = 1;

while (ifeof != EOF) {

ifeof = c = fgetc(map);

if (c == '\n') {

maxy++;

}

}

maxx /= maxy;

fseek(map, 0, SEEK\_SET);

for (int i = 0; i < maxy; i++) {

for (int j = 0; j < maxx; j++) {

fscanf(map, "%d ", &in);

if (in > 0) {

index++;

}

}

}

point\* p = new point[index];

index = 0;

int index7 = 0;

fseek(map, 0, SEEK\_SET);

double START, END;

START = clock();

for (int i = 0; i < maxy; i++) {

for (int j = 0; j < maxx; j++) {

fscanf(map, "%d ", &in);

if (in > 0) {

p[index].x = j;

p[index++].y = i;

}

}

}

printf("充電站位置：\n");

for (int i = 0; i < index; i++) {

printf("(%.0f,%.0f)\n", p[i].x, p[i].y);

}

int index2 = 0, index4 = 0, index5 = 0, index6 = 0;

int index3 = index \* (index - 1) \* (index - 2) / 6;

bline beachline = NULL;

v\* vt = new v[index3];

bv\* bvt = new bv[index3];

e\*\* ev = new e \* [index3];

be\*\* bev = new be \* [index3];

index3 = 0;

while (index != 0) {

if (index3 != 0 && etop(ev)->y <= ptop(p).y && etop(ev)->y <= betop(bev)->y) {

process\_event(ev, &index3, vt, &index4, bev, &index5, bvt, &index6);

}

else if (index5 == 0 || ptop(p).y <= betop(bev)->y) {

process\_point(p, &index, &beachline, ev, &index3, bev, &index5);

}

else if (index5 != 0) {

process\_bevent(bev, &index5, bvt, &index6);

}

}

while (index3 != 0 || index5 != 0) {

if (index5 == 0 || (index3 != 0 && etop(ev)->y <= betop(bev)->y)) {

process\_event(ev, &index3, vt, &index4, bev, &index5, bvt, &index6);

}

else if (index5 != 0) {

process\_bevent(bev, &index5, bvt, &index6);

}

}

freeb(&beachline);

printf("點為：\n");

for (int i = 0; i < index4; i++) {

printf("(%.2f,%.2f)\n", vt[i].p.x, vt[i].p.y);

}

printf("邊界點為：\n");

for (int i = 0; i < index6; i++) {

printf("(%.2f,%.2f)\n", bvt[i].p.x, bvt[i].p.y);

}

print(bvt, index6, vt, index4);

END = clock();

printf("Time:%.0fms", (END - START));

delete[]ev;

delete[]bvt;

delete[]bev;

delete[]vt;

delete[]p;

fclose(map);

system("pause");

return 0;

}

void freeb(bline\* b) {

if (\*b == NULL) {

return;

}

freeb(&((\*b)->rp));

free(\*b);

}

void print(bv\* bvt, int index6, v\* vt, int index4) {

bool ife = true;

for (int i = 0; i < index6; i++) {

if (bvt[i].p.x == 0 && bvt[i].p.y == 0) {

ife = false;

}

}

for (int i = 0; i < index4; i++) {

if (vt[i].p.x == 0 && vt[i].p.y == 0) {

ife = false;

}

}

if (ife) {

printf("(0.00,0.00)\n");

}

ife = true;

for (int i = 0; i < index6; i++) {

if (bvt[i].p.x == 0 && bvt[i].p.y == maxy) {

ife = false;

}

}

for (int i = 0; i < index4; i++) {

if (vt[i].p.x == 0 && vt[i].p.y == maxy) {

ife = false;

}

}

if (ife) {

printf("(0.00,%d.00)\n", maxy);

}

ife = true;

for (int i = 0; i < index6; i++) {

if (bvt[i].p.x == maxx && bvt[i].p.y == 0) {

ife = false;

}

}

for (int i = 0; i < index4; i++) {

if (vt[i].p.x == maxx && vt[i].p.y == 0) {

ife = false;

}

}

if (ife) {

printf("(%d.00,0.00)\n", maxx);

}

ife = true;

for (int i = 0; i < index6; i++) {

if (bvt[i].p.x == maxx && bvt[i].p.y == maxy) {

ife = false;

}

}

for (int i = 0; i < index4; i++) {

if (vt[i].p.x == maxx && vt[i].p.y == maxy) {

ife = false;

}

}

if (ife) {

printf("(%d.00,%d.00)\n", maxx, maxy);

}

}

point ppop(point\* p, int\* index) {

point p0 = p[0];

for (int i = 0; i < \*index - 1; i++) {

p[i] = p[i + 1];

}

(\*index)--;

return p0;

}

point ptop(point\* p) {

return p[0];

}

e\* epop(e\*\* ev, int\* index) {

e\* e0 = ev[0];

for (int i = 0; i < \*index - 1; i++) {

ev[i] = ev[i + 1];

}

(\*index)--;

return e0;

}

e\* etop(e\*\* ev) {

return ev[0];

}

be\* bepop(be\*\* bev, int\* index) {

be\* be0 = bev[0];

for (int i = 0; i < \*index - 1; i++) {

bev[i] = bev[i + 1];

}

(\*index)--;

return be0;

}

be\* betop(be\*\* bev) {

return bev[0];

}

void d(bline b) {

if (b->bev) {

b->bev->valid = false;

b->bev = NULL;

}

if (b->lp->bev) {

b->lp->bev->valid = false;

b->lp->bev = NULL;

}

if (b->bevd) {

b->bevd->valid = false;

b->bevd = NULL;

}

if (b->lp->bevd) {

b->lp->bevd->valid = false;

b->lp->bevd = NULL;

}

if (b->bevr) {

b->bevr->valid = false;

b->bevr = NULL;

}

if (b->lp->bevr) {

b->lp->bevr->valid = false;

b->lp->bevr = NULL;

}

if (b->bevl) {

b->bevl->valid = false;

b->bevl = NULL;

}

if (b->lp->bevl) {

b->lp->bevl->valid = false;

b->lp->bevl = NULL;

}

if (b->ev) {

b->ev->valid = false;

b->ev = NULL;

}

if (b->lp->ev) {

b->lp->ev->valid = false;

b->lp->ev = NULL;

}

if (b->rp->ev) {

b->rp->ev->valid = false;

b->rp->ev = NULL;

}

if (b->rp && b->lp) {

b->rp->lp = b->lp;

b->lp->rp = b->rp;

}

else if (b->rp) {

b->rp->lp = NULL;

}

else if (b->lp) {

b->lp->rp = NULL;

}

}

void process\_point(point\* p, int\* index, bline\* b, e\*\* ev, int\* index3, be\*\* bev, int\* index5) {

point p0 = ppop(p, index);

insert(p0, b, ev, index3, bev, index5);

}

void process\_event(e\*\* equeue, int\* index3, v\* v, int\* index4, be\*\* bequeue, int\* index5, bv\* bvt, int\* index6) {

e\* ev = epop(equeue, index3);

if (ev->valid) {

if (ev->o.x > maxx || ev->o.x < 0 || ev->o.y > maxy || ev->o.y < 0) {

}

else {

v[\*index4].p = ev->o;

v[\*index4].p2 = ev->b->p;

v[\*index4].p1 = ev->b->lp->p;

v[(\*index4)++].p3 = ev->b->rp->p;

}

d(ev->b);

boundu\_event(&ev->b->lp, bequeue, index5, ev->y);

boundd\_event(&ev->b->lp, bequeue, index5, ev->y);

boundr\_event(&ev->b->lp, bequeue, index5, ev->y);

boundl\_event(&ev->b->lp, bequeue, index5, ev->y);

circle\_event(&ev->b->lp, ev->y, equeue, index3);

circle\_event(&ev->b->rp, ev->y, equeue, index3);

if (ev->b->lp) {

free(ev->b);

}

else if (!ev->b->lp && ev->b->rp) {

ev->b = ev->b->rp;

}

else {

ev->b = NULL;

}

}

free(ev);

}

void process\_bevent(be\*\* bequeue, int\* index5, bv\* bv, int\* index6) {

be\* bev = bepop(bequeue, index5);

if (bev->valid) {

bv[\*index6].p = bev->o;

bv[\*index6].p2 = bev->b1->p;

bv[(\*index6)++].p1 = bev->b2->p;

if (bev->o.y == 0) {

bev->b1->bev = NULL;

}

else if (bev->o.y == maxy) {

bev->b1->bevd = NULL;

}

else if (bev->o.x == 0) {

bev->b1->bevl = NULL;

}

else if (bev->o.x == maxx) {

bev->b1->bevr = NULL;

}

}

free(bev);

}

void boundu\_event(bline\* b, be\*\* bequeue, int\* index5, double y0) {

if (!\*b || !(\*b)->rp) {

return;

}

double y;

point o;

be\* bev;

if (boundu((\*b)->p, (\*b)->rp->p, &y, &o) && y >= y0) {

if (o.x <= (\*b)->p.x) {

if (!(\*b)->lp || intersection((\*b)->lp->p, (\*b)->rp->p, y).y < o.y) {

}

else {

return;

}

}

else if (o.x >= (\*b)->rp->p.x) {

if (!(\*b)->rp->rp || intersection((\*b)->p, (\*b)->rp->rp->p, y).y < o.y) {

}

else {

return;

}

}

bev = (be\*)malloc(sizeof(\*bev));

bev->b1 = \*b;

bev->b2 = (\*b)->rp;

bev->o = o;

bev->valid = true;

bev->y = y;

(\*b)->bev = bev;

insertbe(bequeue, bev, index5);

}

}

void boundd\_event(bline\* b, be\*\* bequeue, int\* index5, double y0) {

if (!\*b || !(\*b)->rp) {

return;

}

double y;

point o;

be\* bev;

if (boundd((\*b)->p, (\*b)->rp->p, &y, &o) && y >= y0) {

bev = (be\*)malloc(sizeof(\*bev));

bev->b1 = \*b;

bev->b2 = (\*b)->rp;

bev->o = o;

bev->valid = true;

bev->y = y;

(\*b)->bevd = bev;

insertbe(bequeue, bev, index5);

}

}

void boundr\_event(bline\* b, be\*\* bequeue, int\* index5, double y0) {

if (!\*b || !(\*b)->rp) {

return;

}

double y;

point o;

be\* bev;

if (boundr((\*b)->p, (\*b)->rp->p, &y, &o) && y >= y0) {

bev = (be\*)malloc(sizeof(\*bev));

bev->b1 = \*b;

bev->b2 = (\*b)->rp;

bev->o = o;

bev->valid = true;

bev->y = y;

(\*b)->bevr = bev;

insertbe(bequeue, bev, index5);

}

}

void boundl\_event(bline\* b, be\*\* bequeue, int\* index5, double y0) {

if (!\*b || !(\*b)->rp) {

return;

}

double y;

point o;

be\* bev;

if (boundl((\*b)->p, (\*b)->rp->p, &y, &o) && y >= y0) {

bev = (be\*)malloc(sizeof(\*bev));

bev->b1 = \*b;

bev->b2 = (\*b)->rp;

bev->o = o;

bev->valid = true;

bev->y = y;

(\*b)->bevl = bev;

insertbe(bequeue, bev, index5);

}

}

bool boundl(point p1, point p2, double\* y, point\* o) {

if (p1.y >= p2.y) {

return false;

}

double oy = ((p1.x \* p1.x + p1.y \* p1.y) - (p2.x \* p2.x + p2.y \* p2.y)) / (2 \* (p1.y - p2.y));

if (oy < maxy && oy > 0) {

double y0 = (2 \* oy + sqrt(4 \* oy \* oy - 4 \* (2 \* oy \* p1.y - p1.x \* p1.x - p1.y \* p1.y))) / 2;

o->x = 0;

o->y = oy;

\*y = y0;

return true;

}

return false;

}

bool boundr(point p1, point p2, double\* y, point\* o) {

if (p1.y <= p2.y) {

return false;

}

double oy = (2 \* maxx \* (p2.x - p1.x) + p1.x \* p1.x + p1.y \* p1.y - p2.x \* p2.x - p2.y \* p2.y) / (2 \* (p1.y - p2.y));

if (oy < maxy && oy > 0) {

double y0 = (2 \* oy + sqrt(4 \* oy \* oy - 4 \* (2 \* oy \* p1.y + 2 \* p1.x \* maxx - maxx \* maxx - p1.x \* p1.x - p1.y \* p1.y))) / 2;

o->x = maxx;

o->y = oy;

\*y = y0;

return true;

}

return false;

}

bool boundd(point p1, point p2, double\* y, point\* o) {

if (p1.x == p2.x) {

return false;

}

double ox = (2 \* maxy \* (p2.y - p1.y) + p1.x \* p1.x + p1.y \* p1.y - p2.x \* p2.x - p2.y \* p2.y) / (2 \* (p1.x - p2.x));

if (ox <= maxy && ox >= 0) {

double y0 = (2 \* maxy + sqrt(4 \* maxy \* maxy - 4 \* (2 \* ox \* p1.x + 2 \* p1.y \* maxy - ox \* ox - p1.x \* p1.x - p1.y \* p1.y))) / 2;

o->x = ox;

o->y = maxy;

\*y = y0;

return true;

}

return false;

}

bool boundu(point p1, point p2, double\* y, point\* o) {

if (p1.x == p2.x) {

return false;

}

double ox = ((p1.x \* p1.x + p1.y \* p1.y) - (p2.x \* p2.x + p2.y \* p2.y)) / (2 \* (p1.x - p2.x));

if (ox <= maxx && ox >= 0) {

double y0 = sqrt(ox \* ox - 2 \* ox \* p1.x + p1.x \* p1.x + p1.y \* p1.y);

o->x = ox;

o->y = 0;

\*y = y0;

return true;

}

return false;

}

void inserte(e\*\* equeue, e\* ev, int\* index3) {

if (\*index3 == 0) {

equeue[0] = ev;

(\*index3)++;

return;

}

if (equeue[0]->y >= ev->y) {

for (int j = \*index3; j > 0; j--) {

equeue[j] = equeue[j - 1];

}

equeue[0] = ev;

(\*index3)++;

return;

}

if (equeue[\*index3 - 1]->y <= ev->y) {

equeue[\*index3] = ev;

(\*index3)++;

return;

}

for (int i = 0; i < \*index3 - 1; i++) {

if (equeue[i]->y <= ev->y && equeue[i + 1]->y >= ev->y) {

for (int j = \*index3; j > i + 1; j--) {

equeue[j] = equeue[j - 1];

}

equeue[i + 1] = ev;

(\*index3)++;

break;

}

}

}

void insertbe(be\*\* bequeue, be\* bev, int\* index5) {

if (\*index5 == 0) {

bequeue[0] = bev;

(\*index5)++;

return;

}

if (bequeue[0]->y >= bev->y) {

for (int j = \*index5; j > 0; j--) {

bequeue[j] = bequeue[j - 1];

}

bequeue[0] = bev;

(\*index5)++;

return;

}

if (bequeue[\*index5 - 1]->y <= bev->y) {

bequeue[\*index5] = bev;

(\*index5)++;

return;

}

for (int i = 0; i < \*index5 - 1; i++) {

if (bequeue[i]->y <= bev->y && bequeue[i + 1]->y >= bev->y) {

for (int j = \*index5; j > i + 1; j--) {

bequeue[j] = bequeue[j - 1];

}

bequeue[i + 1] = bev;

(\*index5)++;

break;

}

}

}

void circle\_event(bline\* b, double y0, e\*\* equeue, int\* index3) {

if (!\*b || !(\*b)->lp || !(\*b)->rp) {

return;

}

double y;

point o;

e\* ev;

if (circle((\*b)->lp->p, (\*b)->p, (\*b)->rp->p, &y, &o) && y >= y0) {

ev = (e\*)malloc(sizeof(\*ev));

ev->valid = true;

ev->y = y;

ev->o = o;

ev->b = \*b;

(\*b)->ev = ev;

inserte(equeue, ev, index3);

}

}

bool circle(point a, point b, point c, double\* y, point\* o) {

if ((b.y - a.y) \* (c.x - a.x) - (c.y - a.y) \* (b.x - a.x) > 0) {

return false;

}

double A = 2 \* (b.x - a.x), B = 2 \* (b.y - a.y),

A2 = 2 \* (c.x - b.x), B2 = 2 \* (c.y - b.y),

C = b.x \* b.x + b.y \* b.y - a.x \* a.x - a.y \* a.y,

C2 = c.x \* c.x + c.y \* c.y - b.x \* b.x - b.y \* b.y,

D = (A \* B2 - A2 \* B);

if (D == 0) {

return false;

}

o->x = (B2 \* C - B \* C2) / D;

o->y = (A \* C2 - A2 \* C) / D;

\*y = o->y + sqrt(pow(a.y - o->y, 2) + pow(a.x - o->x, 2));

return true;

}

void insertion(point p, bline\* b) {

if ((\*b)->bev) {

(\*b)->bev->valid = false;

(\*b)->bev = NULL;

}

if ((\*b)->bevd) {

(\*b)->bevd->valid = false;

(\*b)->bevd = NULL;

}

if ((\*b)->bevr) {

(\*b)->bevr->valid = false;

(\*b)->bevr = NULL;

}

if ((\*b)->bevl) {

(\*b)->bevl->valid = false;

(\*b)->bevl = NULL;

}

if ((\*b)->ev) {

(\*b)->ev->valid = false;

(\*b)->ev = NULL;

}

if ((\*b)->rp && (\*b)->rp->ev) {

(\*b)->rp->ev->valid = false;

(\*b)->rp->ev = NULL;

}

bline i, j = \*b;

i = (bline)malloc(sizeof(\*i));

i->p = p;

i->lp = i->rp = NULL;

i->ev = NULL;

i->bev = NULL;

i->bevd = NULL;

i->bevr = NULL;

i->bevl = NULL;

i->lp = j;

if ((\*b)->rp) {

i->rp = j->rp;

j->rp->lp = i;

}

j->rp = i;

}

void insert(point p, bline\* b, e\*\* ev, int\* index3, be\*\* bev, int\* index5) {

if (!\*b) {

bline a = NULL;

a = (bline)malloc(sizeof(\*a));

a->p = p;

a->lp = a->rp = NULL;

a->ev = NULL;

a->bev = NULL;

a->bevd = NULL;

a->bevr = NULL;

a->bevl = NULL;

\*b = a;

return;

}

bool iftwo = false;

int j = 0;

for (bline i = \*b; i; i = i->rp, j++) {

if (ifintersect(p, i)) {

if (j == 0) {

if (intersection(p, i->p, p.y).y > 0) {

bline a;

a = (bline)malloc(sizeof(\*a));

a->p = i->p;

a->lp = a->rp = NULL;

a->ev = NULL;

a->bev = NULL;

a->bevd = NULL;

a->bevr = NULL;

a->bevl = NULL;

a->rp = i;

i->lp = a;

insertion(p, &i->lp);

boundu\_event(&i->lp, bev, index5, p.y);

boundu\_event(&i->lp->lp, bev, index5, p.y);

boundr\_event(&i->lp, bev, index5, p.y);

boundr\_event(&i->lp->lp, bev, index5, p.y);

boundl\_event(&i->lp, bev, index5, p.y);

boundl\_event(&i->lp->lp, bev, index5, p.y);

boundd\_event(&i->lp, bev, index5, p.y);

boundd\_event(&i->lp->lp, bev, index5, p.y);

circle\_event(&i, p.y, ev, index3);

circle\_event(&i->lp, p.y, ev, index3);

\*b = i->lp->lp;

}

else if (intersection(p, i->p, p.y).x > i->p.x) {

insertion(p, &i);

boundu\_event(&i, bev, index5, p.y);

boundu\_event(&i->rp, bev, index5, p.y);

boundd\_event(&i, bev, index5, p.y);

boundd\_event(&i->rp, bev, index5, p.y);

boundr\_event(&i, bev, index5, p.y);

boundr\_event(&i->rp, bev, index5, p.y);

boundl\_event(&i, bev, index5, p.y);

boundl\_event(&i->rp, bev, index5, p.y);

circle\_event(&i->rp, p.y, ev, index3);

circle\_event(&i->rp->rp, p.y, ev, index3);

}

else {

bline a;

a = (bline)malloc(sizeof(\*a));

a->p = p;

a->lp = a->rp = NULL;

a->ev = NULL;

a->bev = NULL;

a->bevd = NULL;

a->bevr = NULL;

a->bevl = NULL;

a->rp = i;

i->lp = a;

boundu\_event(&i->lp, bev, index5, p.y);

boundd\_event(&i->lp, bev, index5, p.y);

boundr\_event(&i->lp, bev, index5, p.y);

boundl\_event(&i->lp, bev, index5, p.y);

circle\_event(&i, p.y, ev, index3);

\*b = i->lp;

}

return;

}

if (!i->rp) {

if (intersection(i->p, p, p.y).y > 0) {

insertion(i->p, &i);

insertion(p, &i);

boundu\_event(&i, bev, index5, p.y);

boundu\_event(&i->rp, bev, index5, p.y);

boundd\_event(&i, bev, index5, p.y);

boundd\_event(&i->rp, bev, index5, p.y);

boundr\_event(&i, bev, index5, p.y);

boundr\_event(&i->rp, bev, index5, p.y);

boundl\_event(&i, bev, index5, p.y);

boundl\_event(&i->rp, bev, index5, p.y);

circle\_event(&i, p.y, ev, index3);

circle\_event(&i->rp, p.y, ev, index3);

}

else if (intersection(p, i->p, p.y).x > i->p.x) {

insertion(p, &i);

boundu\_event(&i, bev, index5, p.y);

boundd\_event(&i, bev, index5, p.y);

boundr\_event(&i, bev, index5, p.y);

boundl\_event(&i, bev, index5, p.y);

circle\_event(&i, p.y, ev, index3);

}

else {

insertion(p, &i->lp);

boundu\_event(&i->lp, bev, index5, p.y);

boundu\_event(&i->lp->lp, bev, index5, p.y);

boundr\_event(&i->lp, bev, index5, p.y);

boundr\_event(&i->lp->lp, bev, index5, p.y);

boundd\_event(&i->lp, bev, index5, p.y);

boundd\_event(&i->lp->lp, bev, index5, p.y);

boundl\_event(&i->lp, bev, index5, p.y);

boundl\_event(&i->lp->lp, bev, index5, p.y);

circle\_event(&i->lp, p.y, ev, index3);

circle\_event(&i->lp->lp, p.y, ev, index3);

}

return;

}

if (i->rp && intersection(p, i->rp->p, p.y).y < intersection(p, i->p, p.y).y) {

insertion(i->p, &i);

iftwo = true;

}

insertion(p, &i);

boundu\_event(&i, bev, index5, p.y);

boundu\_event(&i->rp, bev, index5, p.y);

boundd\_event(&i, bev, index5, p.y);

boundd\_event(&i->rp, bev, index5, p.y);

boundr\_event(&i, bev, index5, p.y);

boundr\_event(&i->rp, bev, index5, p.y);

boundl\_event(&i, bev, index5, p.y);

boundl\_event(&i->rp, bev, index5, p.y);

circle\_event(&i->rp, p.y, ev, index3);

circle\_event(&i, p.y, ev, index3);

circle\_event(&i->rp->rp, p.y, ev, index3);

if (iftwo) {

boundu\_event(&i->rp->rp, bev, index5, p.y);

boundd\_event(&i->rp->rp, bev, index5, p.y);

boundl\_event(&i->rp->rp, bev, index5, p.y);

boundr\_event(&i->rp->rp, bev, index5, p.y);

circle\_event(&i->rp->rp->rp, p.y, ev, index3);

}

return;

}

}

insertion(p, b);

boundu\_event(b, bev, index5, p.y);

boundr\_event(b, bev, index5, p.y);

boundl\_event(b, bev, index5, p.y);

boundd\_event(b, bev, index5, p.y);

}

bool ifintersect(point p1, bline p2) {

if (p2->p.y == p1.y) {

return false;

}

double a, b;

if (p2->lp) {

a = intersection(p2->lp->p, p2->p, p1.y).x;

}

if (p2->rp) {

b = intersection(p2->p, p2->rp->p, p1.y).x;

}

if ((!p2->lp || a <= p1.x) && (!p2->rp || p1.x <= b)) {

return true;

}

return false;

}

point intersection(point p1, point p2, double d) {//找斷點

point p;

if (p1.y == p2.y) {

p.x = (p1.x + p2.x) / 2;

}

else if (p2.y == d) {

p.x = p2.x;

}

else if (p1.y == d) {

p.x = p1.x;

p1 = p2;

}

else {

double z0 = 2 \* (p1.y - d);

double z1 = 2 \* (p2.y - d);

double a = 1 / z0 - 1 / z1;

double b = -2 \* (p1.x / z0 - p2.x / z1);

double c = (p1.x \* p1.x + p1.y \* p1.y - d \* d) / z0 - (p2.x \* p2.x + p2.y \* p2.y - d \* d) / z1;

double x1 = (-b - sqrt(b \* b - 4 \* a \* c)) / (2 \* a);

double x2 = (-b + sqrt(b \* b - 4 \* a \* c)) / (2 \* a);

if (p1.x == p2.x) {

if (p1.y < p2.y) {

if (x1 < x2) {

p.x = x1;

}

else {

p.x = x2;

}

}

else {

if (x1 < x2) {

p.x = x2;

}

else {

p.x = x1;

}

}

}

else if (p1.x > p2.x) {

if (p1.y < p2.y) {

if (x1 < x2) {

p.x = x1;

}

else {

p.x = x2;

}

}

else {

if (x1 < x2) {

p.x = x2;

}

else {

p.x = x1;

}

}

}

else {

if (p1.y < p2.y) {

if (x1 < x2) {

p.x = x1;

}

else {

p.x = x2;

}

}

else {

if (x1 < x2) {

p.x = x2;

}

else {

p.x = x1;

}

}

}

}

p.y = (p1.y \* p1.y + (p1.x - p.x) \* (p1.x - p.x) - d \* d) / (2 \* p1.y - 2 \* d);

return p;

}

2. Input Code Format

Three of examples for input use are in below….

\*因字數過大佔篇幅，所以放在檔案中\*

(1)map1.txt

(2)map2.txt

(3)map3.txt

3. Output Code Format

Three of examples for output use are in below….

\*因字數過大佔篇幅，所以放在檔案中\*

(1)o1.txt

(2)o2.txt

(3)o3.txt

**(2) 依序列出充電站的服務範圍(面積)。**

1. 主程式

#define \_CRT\_SECURE\_NO\_WARNINGS

#include "math.h"

#include "stdio.h"

#include "stdlib.h"

#include "time.h"

int maxx = 0;

int maxy = 0;

struct point {

double x;

double y;

};

struct e;

struct be;

typedef struct bl\* bline;

typedef struct bl {

bline lp, rp;

point p;

e\* ev;

be\* bev;

be\* bevl;

be\* bevr;

be\* bevd;

};

struct e {

double y;

bline b;

point o;

bool valid = true;

};

struct be {

double y;

bline b1, b2;

point o;

bool valid;

};

struct v {

point p;

point p1, p2, p3;

};

struct bv {

point p;

point p1,p2;

};

point ppop(point\* p, int\* index);

point ptop(point\* p);

e\* epop(e\*\* ev, int\* index);

e\* etop(e\*\* ev);

void d(bline b);

void process\_point(point\* p, int\* index, bline\* b, e\*\* ev, int\* index3, be\*\* bev, int\* index5);

void process\_event(e\*\* equeue, int\* index3, v\* v, int\* index4, be\*\* bev, int\* index5, bv\* bvt, int\* index6);

void circle\_event(bline\* b, double y0, e\*\* equeue, int\* index3);

bool circle(point a, point b, point c, double\* y, point\* o);

void insertion(point p, bline\* b);

void insert(point p, bline\* b, e\*\* equeue, int\* index3, be\*\* bequeue, int\* index5);

bool ifintersect(point p1, bline p2);

point intersection(point p1, point p2, double d);

void insertbe(be\*\* bequeue, be\* bev, int\* index5);

be\* bepop(be\*\* bev, int\* index);

be\* betop(be\*\* bev);

bool boundl(point p1, point p2, double\* y, point\* o);

bool boundr(point p1, point p2, double\* y, point\* o);

bool boundu(point p1, point p2, double\* y, point\* o);

bool boundd(point p1, point p2, double\* y, point\* o);

void boundu\_event(bline\* b, be\*\* bequeue, int\* index5, double y0);

void process\_bevent(be\*\* bequeue, int\* index5, bv\* bv, int\* index6);

void boundu\_event(bline\* b, be\*\* bequeue, int\* index5, double y0);

void boundd\_event(bline\* b, be\*\* bequeue, int\* index5, double y0);

void boundr\_event(bline\* b, be\*\* bequeue, int\* index5, double y0);

void boundl\_event(bline\* b, be\*\* bequeue, int\* index5, double y0);

bool boundl(point p1, point p2, double\* y, point\* o);

bool boundr(point p1, point p2, double\* y, point\* o);

bool boundd(point p1, point p2, double\* y, point\* o);

bool boundu(point p1, point p2, double\* y, point\* o);

void inserte(e\*\* equeue, e\* ev, int\* index3);

void print(bv\* bvt, int index6, v\* vt, int index4);

void area(point\* p, int index7, v\* vt, int index4, bv\* bvt, int index6);

void freeb(bline\* b);

int main(void) {

FILE\* map;

char s[10];

printf("請輸入檔案名稱(9位元內)>>");

scanf("%s", s);

map = fopen(s, "r");

int in;

char c;

int index = 0;

int ifeof = 1;

while (ifeof != EOF) {

ifeof = fscanf(map, "%d ", &in);

maxx++;

}

maxx--;

fseek(map, 0, SEEK\_SET);

ifeof = 1;

while (ifeof != EOF) {

ifeof = c = fgetc(map);

if (c == '\n') {

maxy++;

}

}

maxx /= maxy;

fseek(map, 0, SEEK\_SET);

for (int i = 0; i < maxy; i++) {

for (int j = 0; j < maxx; j++) {

fscanf(map, "%d ", &in);

if (in > 0) {

index++;

}

}

}

point\* p = new point[index], \* p2 = new point[index];

index = 0;

int index7 = 0;

fseek(map, 0, SEEK\_SET);

double START, END;

START = clock();

for (int i = 0; i < maxy; i++) {

for (int j = 0; j < maxx; j++) {

fscanf(map, "%d ", &in);

if (in > 0) {

p[index].x = j;

p[index++].y = i;

p2[index7].x = j;

p2[index7++].y = i;

}

}

}

printf("充電站位置：\n");

for (int i = 0; i < index; i++) {

printf("(%.0f,%.0f)\n", p[i].x, p[i].y);

}

int index2 = 0, index4 = 0, index5 = 0, index6 = 0;

int index3 = index \* (index - 1) \* (index - 2) / 6;

bline beachline = NULL;

v\* vt = new v[index3];

bv\* bvt = new bv[index3];

e\*\* ev = new e \* [index3];

be\*\* bev = new be \* [index3];

index3 = 0;

while (index != 0) {

if (index3 != 0 && etop(ev)->y <= ptop(p).y && etop(ev)->y <= betop(bev)->y) {

process\_event(ev, &index3, vt, &index4, bev, &index5, bvt, &index6);

}

else if (index5 == 0 || ptop(p).y <= betop(bev)->y) {

process\_point(p, &index, &beachline, ev, &index3, bev, &index5);

}

else if (index5 != 0) {

process\_bevent(bev, &index5, bvt, &index6);

}

}

while (index3 != 0 || index5 != 0) {

if (index5 == 0 || (index3 != 0 && etop(ev)->y <= betop(bev)->y)) {

process\_event(ev, &index3, vt, &index4, bev, &index5, bvt, &index6);

}

else if (index5 != 0) {

process\_bevent(bev, &index5, bvt, &index6);

}

}

printf("點為：\n");

for (int i = 0; i < index4; i++) {

printf("(%.2f,%.2f)\n", vt[i].p.x, vt[i].p.y);

}

printf("邊界點為：\n");

for (int i = 0; i < index6; i++) {

printf("(%.2f,%.2f)\n", bvt[i].p.x, bvt[i].p.y);

}

print(bvt, index6, vt, index4);

area(p2, index7, vt, index4, bvt, index6);

freeb(&beachline);

END = clock();

printf("Time:%.0fms", (END - START));

delete[]ev;

delete[]bvt;

delete[]bev;

delete[]vt;

delete[]p;

delete[]p2;

fclose(map);

system("pause");

return 0;

}

void print(bv\* bvt, int index6, v\* vt, int index4) {

bool ife = true;

for (int i = 0; i < index6; i++) {

if (bvt[i].p.x == 0 && bvt[i].p.y == 0) {

ife = false;

}

}

for (int i = 0; i < index4; i++) {

if (vt[i].p.x == 0 && vt[i].p.y == 0) {

ife = false;

}

}

if (ife) {

printf("(0.00,0.00)\n");

}

ife = true;

for (int i = 0; i < index6; i++) {

if (bvt[i].p.x == 0 && bvt[i].p.y == maxy) {

ife = false;

}

}

for (int i = 0; i < index4; i++) {

if (vt[i].p.x == 0 && vt[i].p.y == maxy) {

ife = false;

}

}

if (ife) {

printf("(0.00,%d.00)\n", maxy);

}

ife = true;

for (int i = 0; i < index6; i++) {

if (bvt[i].p.x == maxx && bvt[i].p.y == 0) {

ife = false;

}

}

for (int i = 0; i < index4; i++) {

if (vt[i].p.x == maxx && vt[i].p.y == 0) {

ife = false;

}

}

if (ife) {

printf("(%d.00,0.00)\n", maxx);

}

ife = true;

for (int i = 0; i < index6; i++) {

if (bvt[i].p.x == maxx && bvt[i].p.y == maxy) {

ife = false;

}

}

for (int i = 0; i < index4; i++) {

if (vt[i].p.x == maxx && vt[i].p.y == maxy) {

ife = false;

}

}

if (ife) {

printf("(%d.00,%d.00)\n", maxx, maxy);

}

}

void area(point\* p, int index7, v\* vt, int index4, bv\* bvt, int index6) {

point\* pv = new point[index4 + index6];

int index = 0;

point\* px0y0 = new point[index7];

int index2 = 0;

point\* px0ym = new point[index7];

int index3 = 0;

point\* pxmy0 = new point[index7];

int index5 = 0;

point\* pxmym = new point[index7];

int index8 = 0;

double min = maxx \* maxx + maxy \* maxy;

double\* ar = new double[index7];

int index9 = 0;

for (int i = 0; i < index7; i++) {

if (p[i].x \* p[i].x + p[i].y \* p[i].y <= min) {

min = p[i].x \* p[i].x + p[i].y \* p[i].y;

}

}

for (int i = 0; i < index7; i++) {

if (p[i].x \* p[i].x + p[i].y \* p[i].y == min) {

px0y0[index2++] = p[i];

}

}

min = maxx \* maxx + maxy \* maxy;

for (int i = 0; i < index7; i++) {

if (p[i].x \* p[i].x + pow(p[i].y - maxy, 2) <= min) {

min = p[i].x \* p[i].x + pow(p[i].y - maxy, 2);

}

}

for (int i = 0; i < index7; i++) {

if (p[i].x \* p[i].x + pow(p[i].y - maxy, 2) == min) {

px0ym[index3++] = p[i];

}

}

min = maxx \* maxx + maxy \* maxy;

for (int i = 0; i < index7; i++) {

if (p[i].y \* p[i].y + pow(p[i].x - maxx, 2) <= min) {

min = p[i].y \* p[i].y + pow(p[i].x - maxx, 2);

}

}

for (int i = 0; i < index7; i++) {

if (p[i].y \* p[i].y + pow(p[i].x - maxx, 2) == min) {

pxmy0[index5++] = p[i];

}

}

min = maxx \* maxx + maxy \* maxy;

for (int i = 0; i < index7; i++) {

if (pow(p[i].x - maxx, 2) + pow(p[i].y - maxy, 2) <= min) {

min = pow(p[i].x - maxx, 2) + pow(p[i].y - maxy, 2);

}

}

for (int i = 0; i < index7; i++) {

if (pow(p[i].x - maxx, 2) + pow(p[i].y - maxy, 2) == min) {

pxmym[index8++] = p[i];

}

}

for (int i = 0; i < index7; i++) {

for (int j = 0; j < index4; j++) {

if ((vt[j].p1.x == p[i].x && vt[j].p1.y == p[i].y) || (vt[j].p2.x == p[i].x && vt[j].p2.y == p[i].y) || (vt[j].p3.x == p[i].x && vt[j].p3.y == p[i].y)) {

pv[index++] = vt[j].p;

}

}

for (int j = 0; j < index6; j++) {

if ((bvt[j].p1.x == p[i].x && bvt[j].p1.y == p[i].y) || (bvt[j].p2.x == p[i].x && bvt[j].p2.y == p[i].y)) {

pv[index++] = bvt[j].p;

}

}

for (int j = 0; j < index2; j++) {

if (px0y0[j].x == p[i].x && px0y0[j].y == p[i].y) {

pv[index].x = 0;

pv[index++].y = 0;

break;

}

}

for (int j = 0; j < index3; j++) {

if (px0ym[j].x == p[i].x && px0ym[j].y == p[i].y) {

pv[index].x = 0;

pv[index++].y = maxy;

break;

}

}

for (int j = 0; j < index5; j++) {

if (pxmy0[j].x == p[i].x && pxmy0[j].y == p[i].y) {

pv[index].x = maxx;

pv[index++].y = 0;

break;

}

}

for (int j = 0; j < index8; j++) {

if (pxmym[j].x == p[i].x && pxmym[j].y == p[i].y) {

pv[index].x = maxx;

pv[index++].y = maxy;

break;

}

}

point center;

center.x = 0;

center.y = 0;

for (int j = 0; j < index; j++) {

center.x += pv[j].x;

center.y += pv[j].y;

}

center.x /= index;

center.y /= index;

for (int j = 0; j < index - 1; j++) {

for (int k = 0; k < index - j - 1; k++) {

if ((pv[k].x - center.x) \* (pv[k + 1].y - center.y) - (pv[k + 1].x - center.x) \* (pv[k].y - center.y) < 0) {

point temp;

temp = pv[k];

pv[k] = pv[k + 1];

pv[k + 1] = temp;

}

}

}

double a = 0;

for (int j = 0; j < index; j++) {

a += pv[j % index].x \* pv[(j + 1) % index].y - pv[(j + 1) % index].x \* pv[j % index].y;

}

a /= 2;

ar[index9++] = a;

index = 0;

}

for (int i = 0; i < index9 - 1; i++) {

for (int j = 0; j < index9 - i - 1; j++) {

if (ar[j + 1] < ar[j]) {

double temp = ar[j];

point t = p[j];

p[j] = p[j + 1];

p[j + 1] = t;

ar[j] = ar[j + 1];

ar[j + 1] = temp;

}

}

}

for (int i = 0; i < index7; i++) {

printf("充電站(%.0f,%.0f)的服務範圍為%f平方公里\n", p[i].x, p[i].y, ar[i]);

}

delete[]ar;

delete[]pv;

delete[]px0y0;

delete[]px0ym;

delete[]pxmy0;

delete[]pxmym;

}

void freeb(bline\* b) {

if (\*b == NULL) {

return;

}

freeb(&((\*b)->rp));

free(\*b);

}

point ppop(point\* p, int\* index) {

point p0 = p[0];

for (int i = 0; i < \*index - 1; i++) {

p[i] = p[i + 1];

}

(\*index)--;

return p0;

}

point ptop(point\* p) {

return p[0];

}

e\* epop(e\*\* ev, int\* index) {

e\* e0 = ev[0];

for (int i = 0; i < \*index - 1; i++) {

ev[i] = ev[i + 1];

}

(\*index)--;

return e0;

}

e\* etop(e\*\* ev) {

return ev[0];

}

be\* bepop(be\*\* bev, int\* index) {

be\* be0 = bev[0];

for (int i = 0; i < \*index - 1; i++) {

bev[i] = bev[i + 1];

}

(\*index)--;

return be0;

}

be\* betop(be\*\* bev) {

return bev[0];

}

void d(bline b) {

if (b->bev) {

b->bev->valid = false;

b->bev = NULL;

}

if (b->lp->bev) {

b->lp->bev->valid = false;

b->lp->bev = NULL;

}

if (b->bevd) {

b->bevd->valid = false;

b->bevd = NULL;

}

if (b->lp->bevd) {

b->lp->bevd->valid = false;

b->lp->bevd = NULL;

}

if (b->bevr) {

b->bevr->valid = false;

b->bevr = NULL;

}

if (b->lp->bevr) {

b->lp->bevr->valid = false;

b->lp->bevr = NULL;

}

if (b->bevl) {

b->bevl->valid = false;

b->bevl = NULL;

}

if (b->lp->bevl) {

b->lp->bevl->valid = false;

b->lp->bevl = NULL;

}

if (b->ev) {

b->ev->valid = false;

b->ev = NULL;

}

if (b->lp->ev) {

b->lp->ev->valid = false;

b->lp->ev = NULL;

}

if (b->rp->ev) {

b->rp->ev->valid = false;

b->rp->ev = NULL;

}

if (b->rp && b->lp) {

b->rp->lp = b->lp;

b->lp->rp = b->rp;

}

else if (b->rp) {

b->rp->lp = NULL;

}

else if (b->lp) {

b->lp->rp = NULL;

}

}

void process\_point(point\* p, int\* index, bline\* b, e\*\* ev, int\* index3, be\*\* bev, int\* index5) {

point p0 = ppop(p, index);

insert(p0, b, ev, index3, bev, index5);

}

void process\_event(e\*\* equeue, int\* index3, v\* v, int\* index4, be\*\* bequeue, int\* index5, bv\* bvt, int\* index6) {

e\* ev = epop(equeue, index3);

if (ev->valid) {

if (ev->o.x > maxx || ev->o.x < 0 || ev->o.y > maxy || ev->o.y < 0) {

}

else {

v[\*index4].p = ev->o;

v[\*index4].p2 = ev->b->p;

v[\*index4].p1 = ev->b->lp->p;

v[(\*index4)++].p3 = ev->b->rp->p;

}

d(ev->b);

boundu\_event(&ev->b->lp, bequeue, index5, ev->y);

boundd\_event(&ev->b->lp, bequeue, index5, ev->y);

boundr\_event(&ev->b->lp, bequeue, index5, ev->y);

boundl\_event(&ev->b->lp, bequeue, index5, ev->y);

circle\_event(&ev->b->lp, ev->y, equeue, index3);

circle\_event(&ev->b->rp, ev->y, equeue, index3);

if (ev->b->lp) {

free(ev->b);

}

else if (!ev->b->lp && ev->b->rp) {

ev->b = ev->b->rp;

}

else {

ev->b = NULL;

}

}

free(ev);

}

void process\_bevent(be\*\* bequeue, int\* index5, bv\* bv, int\* index6) {

be\* bev = bepop(bequeue, index5);

if (bev->valid) {

bv[\*index6].p = bev->o;

bv[\*index6].p2 = bev->b1->p;

bv[(\*index6)++].p1 = bev->b2->p;

if (bev->o.y == 0) {

bev->b1->bev = NULL;

}

else if (bev->o.y == maxy) {

bev->b1->bevd = NULL;

}

else if (bev->o.x == 0) {

bev->b1->bevl = NULL;

}

else if (bev->o.x == maxx) {

bev->b1->bevr = NULL;

}

}

free(bev);

}

void boundu\_event(bline\* b, be\*\* bequeue, int\* index5, double y0) {

if (!\*b || !(\*b)->rp) {

return;

}

double y;

point o;

be\* bev;

if (boundu((\*b)->p, (\*b)->rp->p, &y, &o) && y >= y0) {

if (o.x <= (\*b)->p.x) {

if (!(\*b)->lp || intersection((\*b)->lp->p, (\*b)->rp->p, y).y < o.y) {

}

else {

return;

}

}

else if (o.x >= (\*b)->rp->p.x) {

if (!(\*b)->rp->rp || intersection((\*b)->p, (\*b)->rp->rp->p, y).y < o.y) {

}

else {

return;

}

}

bev = (be\*)malloc(sizeof(\*bev));

bev->b1 = \*b;

bev->b2 = (\*b)->rp;

bev->o = o;

bev->valid = true;

bev->y = y;

(\*b)->bev = bev;

insertbe(bequeue, bev, index5);

}

}

void boundd\_event(bline\* b, be\*\* bequeue, int\* index5, double y0) {

if (!\*b || !(\*b)->rp) {

return;

}

double y;

point o;

be\* bev;

if (boundd((\*b)->p, (\*b)->rp->p, &y, &o) && y >= y0) {

bev = (be\*)malloc(sizeof(\*bev));

bev->b1 = \*b;

bev->b2 = (\*b)->rp;

bev->o = o;

bev->valid = true;

bev->y = y;

(\*b)->bevd = bev;

insertbe(bequeue, bev, index5);

}

}

void boundr\_event(bline\* b, be\*\* bequeue, int\* index5, double y0) {

if (!\*b || !(\*b)->rp) {

return;

}

double y;

point o;

be\* bev;

if (boundr((\*b)->p, (\*b)->rp->p, &y, &o) && y >= y0) {

bev = (be\*)malloc(sizeof(\*bev));

bev->b1 = \*b;

bev->b2 = (\*b)->rp;

bev->o = o;

bev->valid = true;

bev->y = y;

(\*b)->bevr = bev;

insertbe(bequeue, bev, index5);

}

}

void boundl\_event(bline\* b, be\*\* bequeue, int\* index5, double y0) {

if (!\*b || !(\*b)->rp) {

return;

}

double y;

point o;

be\* bev;

if (boundl((\*b)->p, (\*b)->rp->p, &y, &o) && y >= y0) {

bev = (be\*)malloc(sizeof(\*bev));

bev->b1 = \*b;

bev->b2 = (\*b)->rp;

bev->o = o;

bev->valid = true;

bev->y = y;

(\*b)->bevl = bev;

insertbe(bequeue, bev, index5);

}

}

bool boundl(point p1, point p2, double\* y, point\* o) {

if (p1.y >= p2.y) {

return false;

}

double oy = ((p1.x \* p1.x + p1.y \* p1.y) - (p2.x \* p2.x + p2.y \* p2.y)) / (2 \* (p1.y - p2.y));

if (oy < maxy && oy > 0) {

double y0 = (2 \* oy + sqrt(4 \* oy \* oy - 4 \* (2 \* oy \* p1.y - p1.x \* p1.x - p1.y \* p1.y))) / 2;

o->x = 0;

o->y = oy;

\*y = y0;

return true;

}

return false;

}

bool boundr(point p1, point p2, double\* y, point\* o) {

if (p1.y <= p2.y) {

return false;

}

double oy = (2 \* maxx \* (p2.x - p1.x) + p1.x \* p1.x + p1.y \* p1.y - p2.x \* p2.x - p2.y \* p2.y) / (2 \* (p1.y - p2.y));

if (oy < maxy && oy > 0) {

double y0 = (2 \* oy + sqrt(4 \* oy \* oy - 4 \* (2 \* oy \* p1.y + 2 \* p1.x \* maxx - maxx \* maxx - p1.x \* p1.x - p1.y \* p1.y))) / 2;

o->x = maxx;

o->y = oy;

\*y = y0;

return true;

}

return false;

}

bool boundd(point p1, point p2, double\* y, point\* o) {

if (p1.x == p2.x) {

return false;

}

double ox = (2 \* maxy \* (p2.y - p1.y) + p1.x \* p1.x + p1.y \* p1.y - p2.x \* p2.x - p2.y \* p2.y) / (2 \* (p1.x - p2.x));

if (ox <= maxy && ox >= 0) {

double y0 = (2 \* maxy + sqrt(4 \* maxy \* maxy - 4 \* (2 \* ox \* p1.x + 2 \* p1.y \* maxy - ox \* ox - p1.x \* p1.x - p1.y \* p1.y))) / 2;

o->x = ox;

o->y = maxy;

\*y = y0;

return true;

}

return false;

}

bool boundu(point p1, point p2, double\* y, point\* o) {

if (p1.x == p2.x) {

return false;

}

double ox = ((p1.x \* p1.x + p1.y \* p1.y) - (p2.x \* p2.x + p2.y \* p2.y)) / (2 \* (p1.x - p2.x));

if (ox <= maxx && ox >= 0) {

double y0 = sqrt(ox \* ox - 2 \* ox \* p1.x + p1.x \* p1.x + p1.y \* p1.y);

o->x = ox;

o->y = 0;

\*y = y0;

return true;

}

return false;

}

void inserte(e\*\* equeue, e\* ev, int\* index3) {

if (\*index3 == 0) {

equeue[0] = ev;

(\*index3)++;

return;

}

if (equeue[0]->y >= ev->y) {

for (int j = \*index3; j > 0; j--) {

equeue[j] = equeue[j - 1];

}

equeue[0] = ev;

(\*index3)++;

return;

}

if (equeue[\*index3 - 1]->y <= ev->y) {

equeue[\*index3] = ev;

(\*index3)++;

return;

}

for (int i = 0; i < \*index3 - 1; i++) {

if (equeue[i]->y <= ev->y && equeue[i + 1]->y >= ev->y) {

for (int j = \*index3; j > i + 1; j--) {

equeue[j] = equeue[j - 1];

}

equeue[i + 1] = ev;

(\*index3)++;

break;

}

}

}

void insertbe(be\*\* bequeue, be\* bev, int\* index5) {

if (\*index5 == 0) {

bequeue[0] = bev;

(\*index5)++;

return;

}

if (bequeue[0]->y >= bev->y) {

for (int j = \*index5; j > 0; j--) {

bequeue[j] = bequeue[j - 1];

}

bequeue[0] = bev;

(\*index5)++;

return;

}

if (bequeue[\*index5 - 1]->y <= bev->y) {

bequeue[\*index5] = bev;

(\*index5)++;

return;

}

for (int i = 0; i < \*index5 - 1; i++) {

if (bequeue[i]->y <= bev->y && bequeue[i + 1]->y >= bev->y) {

for (int j = \*index5; j > i + 1; j--) {

bequeue[j] = bequeue[j - 1];

}

bequeue[i + 1] = bev;

(\*index5)++;

break;

}

}

}

void circle\_event(bline\* b, double y0, e\*\* equeue, int\* index3) {

if (!\*b || !(\*b)->lp || !(\*b)->rp) {

return;

}

double y;

point o;

e\* ev;

if (circle((\*b)->lp->p, (\*b)->p, (\*b)->rp->p, &y, &o) && y >= y0) {

ev = (e\*)malloc(sizeof(\*ev));

ev->valid = true;

ev->y = y;

ev->o = o;

ev->b = \*b;

(\*b)->ev = ev;

inserte(equeue, ev, index3);

}

}

bool circle(point a, point b, point c, double\* y, point\* o) {

if ((b.y - a.y) \* (c.x - a.x) - (c.y - a.y) \* (b.x - a.x) > 0) {

return false;

}

double A = 2 \* (b.x - a.x), B = 2 \* (b.y - a.y),

A2 = 2 \* (c.x - b.x), B2 = 2 \* (c.y - b.y),

C = b.x \* b.x + b.y \* b.y - a.x \* a.x - a.y \* a.y,

C2 = c.x \* c.x + c.y \* c.y - b.x \* b.x - b.y \* b.y,

D = (A \* B2 - A2 \* B);

if (D == 0) {

return false;

}

o->x = (B2 \* C - B \* C2) / D;

o->y = (A \* C2 - A2 \* C) / D;

\*y = o->y + sqrt(pow(a.y - o->y, 2) + pow(a.x - o->x, 2));

return true;

}

void insertion(point p, bline\* b) {

if ((\*b)->bev) {

(\*b)->bev->valid = false;

(\*b)->bev = NULL;

}

if ((\*b)->bevd) {

(\*b)->bevd->valid = false;

(\*b)->bevd = NULL;

}

if ((\*b)->bevr) {

(\*b)->bevr->valid = false;

(\*b)->bevr = NULL;

}

if ((\*b)->bevl) {

(\*b)->bevl->valid = false;

(\*b)->bevl = NULL;

}

if ((\*b)->ev) {

(\*b)->ev->valid = false;

(\*b)->ev = NULL;

}

if ((\*b)->rp && (\*b)->rp->ev) {

(\*b)->rp->ev->valid = false;

(\*b)->rp->ev = NULL;

}

bline i, j = \*b;

i = (bline)malloc(sizeof(\*i));

i->p = p;

i->lp = i->rp = NULL;

i->ev = NULL;

i->bev = NULL;

i->bevd = NULL;

i->bevr = NULL;

i->bevl = NULL;

i->lp = j;

if ((\*b)->rp) {

i->rp = j->rp;

j->rp->lp = i;

}

j->rp = i;

}

void insert(point p, bline\* b, e\*\* ev, int\* index3, be\*\* bev, int\* index5) {

if (!\*b) {

bline a = NULL;

a = (bline)malloc(sizeof(\*a));

a->p = p;

a->lp = a->rp = NULL;

a->ev = NULL;

a->bev = NULL;

a->bevd = NULL;

a->bevr = NULL;

a->bevl = NULL;

\*b = a;

return;

}

bool iftwo = false;

int j = 0;

for (bline i = \*b; i; i = i->rp, j++) {

if (ifintersect(p, i)) {

if (j == 0) {

if (intersection(p, i->p, p.y).y > 0) {

bline a;

a = (bline)malloc(sizeof(\*a));

a->p = i->p;

a->lp = a->rp = NULL;

a->ev = NULL;

a->bev = NULL;

a->bevd = NULL;

a->bevr = NULL;

a->bevl = NULL;

a->rp = i;

i->lp = a;

insertion(p, &i->lp);

boundu\_event(&i->lp, bev, index5, p.y);

boundu\_event(&i->lp->lp, bev, index5, p.y);

boundr\_event(&i->lp, bev, index5, p.y);

boundr\_event(&i->lp->lp, bev, index5, p.y);

boundl\_event(&i->lp, bev, index5, p.y);

boundl\_event(&i->lp->lp, bev, index5, p.y);

boundd\_event(&i->lp, bev, index5, p.y);

boundd\_event(&i->lp->lp, bev, index5, p.y);

circle\_event(&i, p.y, ev, index3);

circle\_event(&i->lp, p.y, ev, index3);

\*b = i->lp->lp;

}

else if (intersection(p, i->p, p.y).x > i->p.x) {

insertion(p, &i);

boundu\_event(&i, bev, index5, p.y);

boundu\_event(&i->rp, bev, index5, p.y);

boundd\_event(&i, bev, index5, p.y);

boundd\_event(&i->rp, bev, index5, p.y);

boundr\_event(&i, bev, index5, p.y);

boundr\_event(&i->rp, bev, index5, p.y);

boundl\_event(&i, bev, index5, p.y);

boundl\_event(&i->rp, bev, index5, p.y);

circle\_event(&i->rp, p.y, ev, index3);

circle\_event(&i->rp->rp, p.y, ev, index3);

}

else {

bline a;

a = (bline)malloc(sizeof(\*a));

a->p = p;

a->lp = a->rp = NULL;

a->ev = NULL;

a->bev = NULL;

a->bevd = NULL;

a->bevr = NULL;

a->bevl = NULL;

a->rp = i;

i->lp = a;

boundu\_event(&i->lp, bev, index5, p.y);

boundd\_event(&i->lp, bev, index5, p.y);

boundr\_event(&i->lp, bev, index5, p.y);

boundl\_event(&i->lp, bev, index5, p.y);

circle\_event(&i, p.y, ev, index3);

\*b = i->lp;

}

return;

}

if (!i->rp) {

if (intersection(i->p, p, p.y).y > 0) {

insertion(i->p, &i);

insertion(p, &i);

boundu\_event(&i, bev, index5, p.y);

boundu\_event(&i->rp, bev, index5, p.y);

boundd\_event(&i, bev, index5, p.y);

boundd\_event(&i->rp, bev, index5, p.y);

boundr\_event(&i, bev, index5, p.y);

boundr\_event(&i->rp, bev, index5, p.y);

boundl\_event(&i, bev, index5, p.y);

boundl\_event(&i->rp, bev, index5, p.y);

circle\_event(&i, p.y, ev, index3);

circle\_event(&i->rp, p.y, ev, index3);

}

else if (intersection(p, i->p, p.y).x > i->p.x) {

insertion(p, &i);

boundu\_event(&i, bev, index5, p.y);

boundd\_event(&i, bev, index5, p.y);

boundr\_event(&i, bev, index5, p.y);

boundl\_event(&i, bev, index5, p.y);

circle\_event(&i, p.y, ev, index3);

}

else {

insertion(p, &i->lp);

boundu\_event(&i->lp, bev, index5, p.y);

boundu\_event(&i->lp->lp, bev, index5, p.y);

boundr\_event(&i->lp, bev, index5, p.y);

boundr\_event(&i->lp->lp, bev, index5, p.y);

boundd\_event(&i->lp, bev, index5, p.y);

boundd\_event(&i->lp->lp, bev, index5, p.y);

boundl\_event(&i->lp, bev, index5, p.y);

boundl\_event(&i->lp->lp, bev, index5, p.y);

circle\_event(&i->lp, p.y, ev, index3);

circle\_event(&i->lp->lp, p.y, ev, index3);

}

return;

}

if (i->rp && intersection(p, i->rp->p, p.y).y < intersection(p, i->p, p.y).y) {

insertion(i->p, &i);

iftwo = true;

}

insertion(p, &i);

boundu\_event(&i, bev, index5, p.y);

boundu\_event(&i->rp, bev, index5, p.y);

boundd\_event(&i, bev, index5, p.y);

boundd\_event(&i->rp, bev, index5, p.y);

boundr\_event(&i, bev, index5, p.y);

boundr\_event(&i->rp, bev, index5, p.y);

boundl\_event(&i, bev, index5, p.y);

boundl\_event(&i->rp, bev, index5, p.y);

circle\_event(&i->rp, p.y, ev, index3);

circle\_event(&i, p.y, ev, index3);

circle\_event(&i->rp->rp, p.y, ev, index3);

if (iftwo) {

boundu\_event(&i->rp->rp, bev, index5, p.y);

boundd\_event(&i->rp->rp, bev, index5, p.y);

boundl\_event(&i->rp->rp, bev, index5, p.y);

boundr\_event(&i->rp->rp, bev, index5, p.y);

circle\_event(&i->rp->rp->rp, p.y, ev, index3);

}

return;

}

}

insertion(p, b);

boundu\_event(b, bev, index5, p.y);

boundr\_event(b, bev, index5, p.y);

boundl\_event(b, bev, index5, p.y);

boundd\_event(b, bev, index5, p.y);

}

bool ifintersect(point p1, bline p2) {

if (p2->p.y == p1.y) {

return false;

}

double a, b;

if (p2->lp) {

a = intersection(p2->lp->p, p2->p, p1.y).x;

}

if (p2->rp) {

b = intersection(p2->p, p2->rp->p, p1.y).x;

}

if ((!p2->lp || a <= p1.x) && (!p2->rp || p1.x <= b)) {

return true;

}

return false;

}

point intersection(point p1, point p2, double d) {

point p;

if (p1.y == p2.y) {

p.x = (p1.x + p2.x) / 2;

}

else if (p2.y == d) {

p.x = p2.x;

}

else if (p1.y == d) {

p.x = p1.x;

p1 = p2;

}

else {

double z0 = 2 \* (p1.y - d);

double z1 = 2 \* (p2.y - d);

double a = 1 / z0 - 1 / z1;

double b = -2 \* (p1.x / z0 - p2.x / z1);

double c = (p1.x \* p1.x + p1.y \* p1.y - d \* d) / z0 - (p2.x \* p2.x + p2.y \* p2.y - d \* d) / z1;

double x1 = (-b - sqrt(b \* b - 4 \* a \* c)) / (2 \* a);

double x2 = (-b + sqrt(b \* b - 4 \* a \* c)) / (2 \* a);

if (p1.x == p2.x) {

if (p1.y < p2.y) {

if (x1 < x2) {

p.x = x1;

}

else {

p.x = x2;

}

}

else {

if (x1 < x2) {

p.x = x2;

}

else {

p.x = x1;

}

}

}

else if (p1.x > p2.x) {

if (p1.y < p2.y) {

if (x1 < x2) {

p.x = x1;

}

else {

p.x = x2;

}

}

else {

if (x1 < x2) {

p.x = x2;

}

else {

p.x = x1;

}

}

}

else {

if (p1.y < p2.y) {

if (x1 < x2) {

p.x = x1;

}

else {

p.x = x2;

}

}

else {

if (x1 < x2) {

p.x = x2;

}

else {

p.x = x1;

}

}

}

}

p.y = (p1.y \* p1.y + (p1.x - p.x) \* (p1.x - p.x) - d \* d) / (2 \* p1.y - 2 \* d);

return p;

}

2. Input Code Format

Three of examples for input use are in below….

\*因字數過大佔篇幅，所以放在檔案中\*

(1)map1.txt

(2)map2.txt

(3)map3.txt

3. Output Code Format

Three of examples for output use are in below….

\*因字數過大佔篇幅，所以放在檔案中\*

(1)o1.txt

(2)o2.txt

(3)o3.txt

**(3) 找出離充電站最遠的點以及距離(Largest Empty Circle Problem with Voronoi Diagram, also called toxic waste dump problem)。**

1. 主程式

#define \_CRT\_SECURE\_NO\_WARNINGS

#include "math.h"

#include "stdio.h"

#include "stdlib.h"

#include "time.h"

int maxx = 0;

int maxy = 0;

struct point {

double x;

double y;

};

struct e;

struct be;

typedef struct bl\* bline;

typedef struct bl {

bline lp, rp;

point p;

e\* ev;

be\* bev;

be\* bevl;

be\* bevr;

be\* bevd;

};

struct e {

double y;

bline b;

point o;

bool valid = true;

};

struct be {

double y;

bline b1, b2;

point o;

bool valid;

};

struct v {

point p;

point p1, p2, p3;

};

struct bv {

point p;

point p1, p2;

};

point ppop(point\* p, int\* index);

point ptop(point\* p);

e\* epop(e\*\* ev, int\* index);

e\* etop(e\*\* ev);

void d(bline b);

void process\_point(point\* p, int\* index, bline\* b, e\*\* ev, int\* index3, be\*\* bev, int\* index5);

void process\_event(e\*\* equeue, int\* index3, v\* v, int\* index4, be\*\* bev, int\* index5, bv\* bvt, int\* index6);

void circle\_event(bline\* b, double y0, e\*\* equeue, int\* index3);

bool circle(point a, point b, point c, double\* y, point\* o);

void insertion(point p, bline\* b);

void insert(point p, bline\* b, e\*\* equeue, int\* index3, be\*\* bequeue, int\* index5);

bool ifintersect(point p1, bline p2);

point intersection(point p1, point p2, double d);

void insertbe(be\*\* bequeue, be\* bev, int\* index5);

be\* bepop(be\*\* bev, int\* index);

be\* betop(be\*\* bev);

bool boundl(point p1, point p2, double\* y, point\* o);

bool boundr(point p1, point p2, double\* y, point\* o);

bool boundu(point p1, point p2, double\* y, point\* o);

bool boundd(point p1, point p2, double\* y, point\* o);

void boundu\_event(bline\* b, be\*\* bequeue, int\* index5, double y0);

void process\_bevent(be\*\* bequeue, int\* index5, bv\* bv, int\* index6);

void boundu\_event(bline\* b, be\*\* bequeue, int\* index5, double y0);

void boundd\_event(bline\* b, be\*\* bequeue, int\* index5, double y0);

void boundr\_event(bline\* b, be\*\* bequeue, int\* index5, double y0);

void boundl\_event(bline\* b, be\*\* bequeue, int\* index5, double y0);

bool boundl(point p1, point p2, double\* y, point\* o);

bool boundr(point p1, point p2, double\* y, point\* o);

bool boundd(point p1, point p2, double\* y, point\* o);

bool boundu(point p1, point p2, double\* y, point\* o);

void inserte(e\*\* equeue, e\* ev, int\* index3);

void print(bv\* bvt, int index6, v\* vt, int index4);

void dt(v\* vt, int index4, bv\* bvt, int index6, point\* p, int index);

void freeb(bline\* b);

int main(void) {

FILE\* map;

char s[10];

printf("請輸入檔案名稱(9位元內)>>");

scanf("%s", s);

map = fopen(s, "r");

int in;

char c;

int index = 0;

int ifeof = 1;

while (ifeof != EOF) {

ifeof = fscanf(map, "%d ", &in);

maxx++;

}

maxx--;

fseek(map, 0, SEEK\_SET);

ifeof = 1;

while (ifeof != EOF) {

ifeof = c = fgetc(map);

if (c == '\n') {

maxy++;

}

}

maxx /= maxy;

fseek(map, 0, SEEK\_SET);

for (int i = 0; i < maxy; i++) {

for (int j = 0; j < maxx; j++) {

fscanf(map, "%d ", &in);

if (in > 0) {

index++;

}

}

}

point\* p = new point[index], \* p2 = new point[index];

index = 0;

int index7 = 0;

fseek(map, 0, SEEK\_SET);

double START, END;

START = clock();

for (int i = 0; i < maxy; i++) {

for (int j = 0; j < maxx; j++) {

fscanf(map, "%d ", &in);

if (in > 0) {

p[index].x = j;

p[index++].y = i;

p2[index7].x = j;

p2[index7++].y = i;

}

}

}

printf("充電站位置：\n");

for (int i = 0; i < index; i++) {

printf("(%.0f,%.0f)\n", p[i].x, p[i].y);

}

int index2 = 0, index4 = 0, index5 = 0, index6 = 0;

int index3 = index \* (index - 1) \* (index - 2) / 6;

bline beachline = NULL;

v\* vt = new v[index3];

bv\* bvt = new bv[index3];

e\*\* ev = new e \* [index3];

be\*\* bev = new be \* [index3];

index3 = 0;

while (index != 0) {

if (index3 != 0 && etop(ev)->y <= ptop(p).y && etop(ev)->y <= betop(bev)->y) {

process\_event(ev, &index3, vt, &index4, bev, &index5, bvt, &index6);

}

else if (index5 == 0 || ptop(p).y <= betop(bev)->y) {

process\_point(p, &index, &beachline, ev, &index3, bev, &index5);

}

else if (index5 != 0) {

process\_bevent(bev, &index5, bvt, &index6);

}

}

while (index3 != 0 || index5 != 0) {

if (index5 == 0 || (index3 != 0 && etop(ev)->y <= betop(bev)->y)) {

process\_event(ev, &index3, vt, &index4, bev, &index5, bvt, &index6);

}

else if (index5 != 0) {

process\_bevent(bev, &index5, bvt, &index6);

}

}

printf("點為：\n");

for (int i = 0; i < index4; i++) {

printf("(%.2f,%.2f)\n", vt[i].p.x, vt[i].p.y);

}

printf("邊界點為：\n");

for (int i = 0; i < index6; i++) {

printf("(%.2f,%.2f)\n", bvt[i].p.x, bvt[i].p.y);

}

print(bvt, index6, vt, index4);

dt(vt, index4, bvt, index6, p2, index7);

END = clock();

printf("Time:%.0fms", (END - START));

delete[]ev;

delete[]bvt;

delete[]bev;

delete[]vt;

delete[]p;

delete[]p2;

fclose(map);

freeb(&beachline);

system("pause");

return 0;

}

void freeb(bline\* b) {

if (\*b == NULL) {

return;

}

freeb(&((\*b)->rp));

free(\*b);

}

void dt(v\* vt, int index4, bv\* bvt, int index6, point\* p, int index) {

double max = 0;

point o;

for (int i = 0; i < index4; i++) {

if (max <= pow(vt[i].p1.x - vt[i].p.x, 2) + pow(vt[i].p1.y - vt[i].p.y, 2)) {

bool ife2 = true;

for (int j = 0; j < index; j++) {

if (vt[i].p.x == p[j].x && vt[i].p.y == p[j].y) {

ife2 = false;

break;

}

}

if (ife2) {

max = pow(vt[i].p1.x - vt[i].p.x, 2) + pow(vt[i].p1.y - vt[i].p.y, 2);

o = vt[i].p;

}

}

}

for (int i = 0; i < index6; i++) {

if (max <= pow(bvt[i].p1.x - bvt[i].p.x, 2) + pow(bvt[i].p1.y - bvt[i].p.y, 2)) {

bool ife2 = true;

for (int j = 0; j < index; j++) {

if (bvt[i].p.x == p[j].x && bvt[i].p.y == p[j].y) {

ife2 = false;

break;

}

}

if (ife2) {

max = pow(bvt[i].p1.x - bvt[i].p.x, 2) + pow(bvt[i].p1.y - bvt[i].p.y, 2);

o = bvt[i].p;

}

}

}

double min = maxx \* maxx + maxy \* maxy;

bool ife = true;

for (int i = 0; i < index; i++) {

if (p[i].x == 0 && p[i].y == 0) {

ife = false;

break;

}

}

if (ife) {

for (int i = 0; i < index; i++) {

if (min >= pow(p[i].x, 2) + pow(p[i].y, 2)) {

min = pow(p[i].x, 2) + pow(p[i].y, 2);

}

}

if (max < min) {

max = min;

o.x = 0;

o.y = 0;

}

}

ife = true;

min = maxx \* maxx + maxy \* maxy;

for (int i = 0; i < index; i++) {

if (p[i].x == 0 && p[i].y == maxy) {

ife = false;

break;

}

}

if (ife) {

for (int i = 0; i < index; i++) {

if (min >= pow(p[i].x, 2) + pow(p[i].y - maxy, 2)) {

min = pow(p[i].x, 2) + pow(p[i].y - maxy, 2);

}

}

if (max < min) {

max = min;

o.x = 0;

o.y = maxy;

}

}

ife = true;

min = maxx \* maxx + maxy \* maxy;

for (int i = 0; i < index; i++) {

if (p[i].x == maxx && p[i].y == maxy) {

ife = false;

break;

}

}

if (ife) {

for (int i = 0; i < index; i++) {

if (min >= pow(p[i].x - maxx, 2) + pow(p[i].y - maxy, 2)) {

min = pow(p[i].x - maxx, 2) + pow(p[i].y - maxy, 2);

}

}

if (max <= min) {

max = min;

o.x = maxx;

o.y = maxy;

}

}

ife = true;

min = maxx \* maxx + maxy \* maxy;

for (int i = 0; i < index; i++) {

if (p[i].x == maxx && p[i].y == 0) {

ife = false;

break;

}

}

if (ife) {

for (int i = 0; i < index; i++) {

if (min >= pow(p[i].x - maxx, 2) + pow(p[i].y, 2)) {

min = pow(p[i].x - maxx, 2) + pow(p[i].y, 2);

}

}

if (max < min) {

max = min;

o.x = maxx;

o.y = 0;

}

}

printf("離充電站最遠的點為(%.2f,%.2f)距離%f公里\n", o.x, o.y, sqrt(max));

}

void print(bv\* bvt, int index6, v\* vt, int index4) {

bool ife = true;

for (int i = 0; i < index6; i++) {

if (bvt[i].p.x == 0 && bvt[i].p.y == 0) {

ife = false;

}

}

for (int i = 0; i < index4; i++) {

if (vt[i].p.x == 0 && vt[i].p.y == 0) {

ife = false;

}

}

if (ife) {

printf("(0.00,0.00)\n");

}

ife = true;

for (int i = 0; i < index6; i++) {

if (bvt[i].p.x == 0 && bvt[i].p.y == maxy) {

ife = false;

}

}

for (int i = 0; i < index4; i++) {

if (vt[i].p.x == 0 && vt[i].p.y == maxy) {

ife = false;

}

}

if (ife) {

printf("(0.00,%d.00)\n", maxy);

}

ife = true;

for (int i = 0; i < index6; i++) {

if (bvt[i].p.x == maxx && bvt[i].p.y == 0) {

ife = false;

}

}

for (int i = 0; i < index4; i++) {

if (vt[i].p.x == maxx && vt[i].p.y == 0) {

ife = false;

}

}

if (ife) {

printf("(%d.00,0.00)\n", maxx);

}

ife = true;

for (int i = 0; i < index6; i++) {

if (bvt[i].p.x == maxx && bvt[i].p.y == maxy) {

ife = false;

}

}

for (int i = 0; i < index4; i++) {

if (vt[i].p.x == maxx && vt[i].p.y == maxy) {

ife = false;

}

}

if (ife) {

printf("(%d.00,%d.00)\n", maxx, maxy);

}

}

point ppop(point\* p, int\* index) {

point p0 = p[0];

for (int i = 0; i < \*index - 1; i++) {

p[i] = p[i + 1];

}

(\*index)--;

return p0;

}

point ptop(point\* p) {

return p[0];

}

e\* epop(e\*\* ev, int\* index) {

e\* e0 = ev[0];

for (int i = 0; i < \*index - 1; i++) {

ev[i] = ev[i + 1];

}

(\*index)--;

return e0;

}

e\* etop(e\*\* ev) {

return ev[0];

}

be\* bepop(be\*\* bev, int\* index) {

be\* be0 = bev[0];

for (int i = 0; i < \*index - 1; i++) {

bev[i] = bev[i + 1];

}

(\*index)--;

return be0;

}

be\* betop(be\*\* bev) {

return bev[0];

}

void d(bline b) {

if (b->bev) {

b->bev->valid = false;

b->bev = NULL;

}

if (b->lp->bev) {

b->lp->bev->valid = false;

b->lp->bev = NULL;

}

if (b->bevd) {

b->bevd->valid = false;

b->bevd = NULL;

}

if (b->lp->bevd) {

b->lp->bevd->valid = false;

b->lp->bevd = NULL;

}

if (b->bevr) {

b->bevr->valid = false;

b->bevr = NULL;

}

if (b->lp->bevr) {

b->lp->bevr->valid = false;

b->lp->bevr = NULL;

}

if (b->bevl) {

b->bevl->valid = false;

b->bevl = NULL;

}

if (b->lp->bevl) {

b->lp->bevl->valid = false;

b->lp->bevl = NULL;

}

if (b->ev) {

b->ev->valid = false;

b->ev = NULL;

}

if (b->lp->ev) {

b->lp->ev->valid = false;

b->lp->ev = NULL;

}

if (b->rp->ev) {

b->rp->ev->valid = false;

b->rp->ev = NULL;

}

if (b->rp && b->lp) {

b->rp->lp = b->lp;

b->lp->rp = b->rp;

}

else if (b->rp) {

b->rp->lp = NULL;

}

else if (b->lp) {

b->lp->rp = NULL;

}

}

void process\_point(point\* p, int\* index, bline\* b, e\*\* ev, int\* index3, be\*\* bev, int\* index5) {

point p0 = ppop(p, index);

insert(p0, b, ev, index3, bev, index5);

}

void process\_event(e\*\* equeue, int\* index3, v\* v, int\* index4, be\*\* bequeue, int\* index5, bv\* bvt, int\* index6) {

e\* ev = epop(equeue, index3);

if (ev->valid) {

if (ev->o.x > maxx || ev->o.x < 0 || ev->o.y > maxy || ev->o.y < 0) {

}

else {

v[\*index4].p = ev->o;

v[\*index4].p2 = ev->b->p;

v[\*index4].p1 = ev->b->lp->p;

v[(\*index4)++].p3 = ev->b->rp->p;

}

d(ev->b);

boundu\_event(&ev->b->lp, bequeue, index5, ev->y);

boundd\_event(&ev->b->lp, bequeue, index5, ev->y);

boundr\_event(&ev->b->lp, bequeue, index5, ev->y);

boundl\_event(&ev->b->lp, bequeue, index5, ev->y);

circle\_event(&ev->b->lp, ev->y, equeue, index3);

circle\_event(&ev->b->rp, ev->y, equeue, index3);

if (ev->b->lp) {

free(ev->b);

}

else if (!ev->b->lp && ev->b->rp) {

ev->b = ev->b->rp;

}

else {

ev->b = NULL;

}

}

free(ev);

}

void process\_bevent(be\*\* bequeue, int\* index5, bv\* bv, int\* index6) {

be\* bev = bepop(bequeue, index5);

if (bev->valid) {

bv[\*index6].p = bev->o;

bv[\*index6].p2 = bev->b1->p;

bv[(\*index6)++].p1 = bev->b2->p;

if (bev->o.y == 0) {

bev->b1->bev = NULL;

}

else if (bev->o.y == maxy) {

bev->b1->bevd = NULL;

}

else if (bev->o.x == 0) {

bev->b1->bevl = NULL;

}

else if (bev->o.x == maxx) {

bev->b1->bevr = NULL;

}

}

free(bev);

}

void boundu\_event(bline\* b, be\*\* bequeue, int\* index5, double y0) {

if (!\*b || !(\*b)->rp) {

return;

}

double y;

point o;

be\* bev;

if (boundu((\*b)->p, (\*b)->rp->p, &y, &o) && y >= y0) {

if (o.x <= (\*b)->p.x) {

if (!(\*b)->lp || intersection((\*b)->lp->p, (\*b)->rp->p, y).y < o.y) {

}

else {

return;

}

}

else if (o.x >= (\*b)->rp->p.x) {

if (!(\*b)->rp->rp || intersection((\*b)->p, (\*b)->rp->rp->p, y).y < o.y) {

}

else {

return;

}

}

bev = (be\*)malloc(sizeof(\*bev));

bev->b1 = \*b;

bev->b2 = (\*b)->rp;

bev->o = o;

bev->valid = true;

bev->y = y;

(\*b)->bev = bev;

insertbe(bequeue, bev, index5);

}

}

void boundd\_event(bline\* b, be\*\* bequeue, int\* index5, double y0) {

if (!\*b || !(\*b)->rp) {

return;

}

double y;

point o;

be\* bev;

if (boundd((\*b)->p, (\*b)->rp->p, &y, &o) && y >= y0) {

bev = (be\*)malloc(sizeof(\*bev));

bev->b1 = \*b;

bev->b2 = (\*b)->rp;

bev->o = o;

bev->valid = true;

bev->y = y;

(\*b)->bevd = bev;

insertbe(bequeue, bev, index5);

}

}

void boundr\_event(bline\* b, be\*\* bequeue, int\* index5, double y0) {

if (!\*b || !(\*b)->rp) {

return;

}

double y;

point o;

be\* bev;

if (boundr((\*b)->p, (\*b)->rp->p, &y, &o) && y >= y0) {

bev = (be\*)malloc(sizeof(\*bev));

bev->b1 = \*b;

bev->b2 = (\*b)->rp;

bev->o = o;

bev->valid = true;

bev->y = y;

(\*b)->bevr = bev;

insertbe(bequeue, bev, index5);

}

}

void boundl\_event(bline\* b, be\*\* bequeue, int\* index5, double y0) {

if (!\*b || !(\*b)->rp) {

return;

}

double y;

point o;

be\* bev;

if (boundl((\*b)->p, (\*b)->rp->p, &y, &o) && y >= y0) {

bev = (be\*)malloc(sizeof(\*bev));

bev->b1 = \*b;

bev->b2 = (\*b)->rp;

bev->o = o;

bev->valid = true;

bev->y = y;

(\*b)->bevl = bev;

insertbe(bequeue, bev, index5);

}

}

bool boundl(point p1, point p2, double\* y, point\* o) {

if (p1.y >= p2.y) {

return false;

}

double oy = ((p1.x \* p1.x + p1.y \* p1.y) - (p2.x \* p2.x + p2.y \* p2.y)) / (2 \* (p1.y - p2.y));

if (oy < maxy && oy > 0) {

double y0 = (2 \* oy + sqrt(4 \* oy \* oy - 4 \* (2 \* oy \* p1.y - p1.x \* p1.x - p1.y \* p1.y))) / 2;

o->x = 0;

o->y = oy;

\*y = y0;

return true;

}

return false;

}

bool boundr(point p1, point p2, double\* y, point\* o) {

if (p1.y <= p2.y) {

return false;

}

double oy = (2 \* maxx \* (p2.x - p1.x) + p1.x \* p1.x + p1.y \* p1.y - p2.x \* p2.x - p2.y \* p2.y) / (2 \* (p1.y - p2.y));

if (oy < maxy && oy > 0) {

double y0 = (2 \* oy + sqrt(4 \* oy \* oy - 4 \* (2 \* oy \* p1.y + 2 \* p1.x \* maxx - maxx \* maxx - p1.x \* p1.x - p1.y \* p1.y))) / 2;

o->x = maxx;

o->y = oy;

\*y = y0;

return true;

}

return false;

}

bool boundd(point p1, point p2, double\* y, point\* o) {

if (p1.x == p2.x) {

return false;

}

double ox = (2 \* maxy \* (p2.y - p1.y) + p1.x \* p1.x + p1.y \* p1.y - p2.x \* p2.x - p2.y \* p2.y) / (2 \* (p1.x - p2.x));

if (ox <= maxy && ox >= 0) {

double y0 = (2 \* maxy + sqrt(4 \* maxy \* maxy - 4 \* (2 \* ox \* p1.x + 2 \* p1.y \* maxy - ox \* ox - p1.x \* p1.x - p1.y \* p1.y))) / 2;

o->x = ox;

o->y = maxy;

\*y = y0;

return true;

}

return false;

}

bool boundu(point p1, point p2, double\* y, point\* o) {

if (p1.x == p2.x) {

return false;

}

double ox = ((p1.x \* p1.x + p1.y \* p1.y) - (p2.x \* p2.x + p2.y \* p2.y)) / (2 \* (p1.x - p2.x));

if (ox <= maxx && ox >= 0) {

double y0 = sqrt(ox \* ox - 2 \* ox \* p1.x + p1.x \* p1.x + p1.y \* p1.y);

o->x = ox;

o->y = 0;

\*y = y0;

return true;

}

return false;

}

void inserte(e\*\* equeue, e\* ev, int\* index3) {

if (\*index3 == 0) {

equeue[0] = ev;

(\*index3)++;

return;

}

if (equeue[0]->y >= ev->y) {

for (int j = \*index3; j > 0; j--) {

equeue[j] = equeue[j - 1];

}

equeue[0] = ev;

(\*index3)++;

return;

}

if (equeue[\*index3 - 1]->y <= ev->y) {

equeue[\*index3] = ev;

(\*index3)++;

return;

}

for (int i = 0; i < \*index3 - 1; i++) {

if (equeue[i]->y <= ev->y && equeue[i + 1]->y >= ev->y) {

for (int j = \*index3; j > i + 1; j--) {

equeue[j] = equeue[j - 1];

}

equeue[i + 1] = ev;

(\*index3)++;

break;

}

}

}

void insertbe(be\*\* bequeue, be\* bev, int\* index5) {

if (\*index5 == 0) {

bequeue[0] = bev;

(\*index5)++;

return;

}

if (bequeue[0]->y >= bev->y) {

for (int j = \*index5; j > 0; j--) {

bequeue[j] = bequeue[j - 1];

}

bequeue[0] = bev;

(\*index5)++;

return;

}

if (bequeue[\*index5 - 1]->y <= bev->y) {

bequeue[\*index5] = bev;

(\*index5)++;

return;

}

for (int i = 0; i < \*index5 - 1; i++) {

if (bequeue[i]->y <= bev->y && bequeue[i + 1]->y >= bev->y) {

for (int j = \*index5; j > i + 1; j--) {

bequeue[j] = bequeue[j - 1];

}

bequeue[i + 1] = bev;

(\*index5)++;

break;

}

}

}

void circle\_event(bline\* b, double y0, e\*\* equeue, int\* index3) {

if (!\*b || !(\*b)->lp || !(\*b)->rp) {

return;

}

double y;

point o;

e\* ev;

if (circle((\*b)->lp->p, (\*b)->p, (\*b)->rp->p, &y, &o) && y >= y0) {

ev = (e\*)malloc(sizeof(\*ev));

ev->valid = true;

ev->y = y;

ev->o = o;

ev->b = \*b;

(\*b)->ev = ev;

inserte(equeue, ev, index3);

}

}

bool circle(point a, point b, point c, double\* y, point\* o) {

if ((b.y - a.y) \* (c.x - a.x) - (c.y - a.y) \* (b.x - a.x) > 0) {

return false;

}

double A = 2 \* (b.x - a.x), B = 2 \* (b.y - a.y),

A2 = 2 \* (c.x - b.x), B2 = 2 \* (c.y - b.y),

C = b.x \* b.x + b.y \* b.y - a.x \* a.x - a.y \* a.y,

C2 = c.x \* c.x + c.y \* c.y - b.x \* b.x - b.y \* b.y,

D = (A \* B2 - A2 \* B);

if (D == 0) {

return false;

}

o->x = (B2 \* C - B \* C2) / D;

o->y = (A \* C2 - A2 \* C) / D;

\*y = o->y + sqrt(pow(a.y - o->y, 2) + pow(a.x - o->x, 2));

return true;

}

void insertion(point p, bline\* b) {

if ((\*b)->bev) {

(\*b)->bev->valid = false;

(\*b)->bev = NULL;

}

if ((\*b)->bevd) {

(\*b)->bevd->valid = false;

(\*b)->bevd = NULL;

}

if ((\*b)->bevr) {

(\*b)->bevr->valid = false;

(\*b)->bevr = NULL;

}

if ((\*b)->bevl) {

(\*b)->bevl->valid = false;

(\*b)->bevl = NULL;

}

if ((\*b)->ev) {

(\*b)->ev->valid = false;

(\*b)->ev = NULL;

}

if ((\*b)->rp && (\*b)->rp->ev) {

(\*b)->rp->ev->valid = false;

(\*b)->rp->ev = NULL;

}

bline i, j = \*b;

i = (bline)malloc(sizeof(\*i));

i->p = p;

i->lp = i->rp = NULL;

i->ev = NULL;

i->bev = NULL;

i->bevd = NULL;

i->bevr = NULL;

i->bevl = NULL;

i->lp = j;

if ((\*b)->rp) {

i->rp = j->rp;

j->rp->lp = i;

}

j->rp = i;

}

void insert(point p, bline\* b, e\*\* ev, int\* index3, be\*\* bev, int\* index5) {

if (!\*b) {

bline a = NULL;

a = (bline)malloc(sizeof(\*a));

a->p = p;

a->lp = a->rp = NULL;

a->ev = NULL;

a->bev = NULL;

a->bevd = NULL;

a->bevr = NULL;

a->bevl = NULL;

\*b = a;

return;

}

bool iftwo = false;

int j = 0;

for (bline i = \*b; i; i = i->rp, j++) {

if (ifintersect(p, i)) {

if (j == 0) {

if (intersection(p, i->p, p.y).y > 0) {

bline a;

a = (bline)malloc(sizeof(\*a));

a->p = i->p;

a->lp = a->rp = NULL;

a->ev = NULL;

a->bev = NULL;

a->bevd = NULL;

a->bevr = NULL;

a->bevl = NULL;

a->rp = i;

i->lp = a;

insertion(p, &i->lp);

boundu\_event(&i->lp, bev, index5, p.y);

boundu\_event(&i->lp->lp, bev, index5, p.y);

boundr\_event(&i->lp, bev, index5, p.y);

boundr\_event(&i->lp->lp, bev, index5, p.y);

boundl\_event(&i->lp, bev, index5, p.y);

boundl\_event(&i->lp->lp, bev, index5, p.y);

boundd\_event(&i->lp, bev, index5, p.y);

boundd\_event(&i->lp->lp, bev, index5, p.y);

circle\_event(&i, p.y, ev, index3);

circle\_event(&i->lp, p.y, ev, index3);

\*b = i->lp->lp;

}

else if (intersection(p, i->p, p.y).x > i->p.x) {

insertion(p, &i);

boundu\_event(&i, bev, index5, p.y);

boundu\_event(&i->rp, bev, index5, p.y);

boundd\_event(&i, bev, index5, p.y);

boundd\_event(&i->rp, bev, index5, p.y);

boundr\_event(&i, bev, index5, p.y);

boundr\_event(&i->rp, bev, index5, p.y);

boundl\_event(&i, bev, index5, p.y);

boundl\_event(&i->rp, bev, index5, p.y);

circle\_event(&i->rp, p.y, ev, index3);

circle\_event(&i->rp->rp, p.y, ev, index3);

}

else {

bline a;

a = (bline)malloc(sizeof(\*a));

a->p = p;

a->lp = a->rp = NULL;

a->ev = NULL;

a->bev = NULL;

a->bevd = NULL;

a->bevr = NULL;

a->bevl = NULL;

a->rp = i;

i->lp = a;

boundu\_event(&i->lp, bev, index5, p.y);

boundd\_event(&i->lp, bev, index5, p.y);

boundr\_event(&i->lp, bev, index5, p.y);

boundl\_event(&i->lp, bev, index5, p.y);

circle\_event(&i, p.y, ev, index3);

\*b = i->lp;

}

return;

}

if (!i->rp) {

if (intersection(i->p, p, p.y).y > 0) {

insertion(i->p, &i);

insertion(p, &i);

boundu\_event(&i, bev, index5, p.y);

boundu\_event(&i->rp, bev, index5, p.y);

boundd\_event(&i, bev, index5, p.y);

boundd\_event(&i->rp, bev, index5, p.y);

boundr\_event(&i, bev, index5, p.y);

boundr\_event(&i->rp, bev, index5, p.y);

boundl\_event(&i, bev, index5, p.y);

boundl\_event(&i->rp, bev, index5, p.y);

circle\_event(&i, p.y, ev, index3);

circle\_event(&i->rp, p.y, ev, index3);

}

else if (intersection(p, i->p, p.y).x > i->p.x) {

insertion(p, &i);

boundu\_event(&i, bev, index5, p.y);

boundd\_event(&i, bev, index5, p.y);

boundr\_event(&i, bev, index5, p.y);

boundl\_event(&i, bev, index5, p.y);

circle\_event(&i, p.y, ev, index3);

}

else {

insertion(p, &i->lp);

boundu\_event(&i->lp, bev, index5, p.y);

boundu\_event(&i->lp->lp, bev, index5, p.y);

boundr\_event(&i->lp, bev, index5, p.y);

boundr\_event(&i->lp->lp, bev, index5, p.y);

boundd\_event(&i->lp, bev, index5, p.y);

boundd\_event(&i->lp->lp, bev, index5, p.y);

boundl\_event(&i->lp, bev, index5, p.y);

boundl\_event(&i->lp->lp, bev, index5, p.y);

circle\_event(&i->lp, p.y, ev, index3);

circle\_event(&i->lp->lp, p.y, ev, index3);

}

return;

}

if (i->rp && intersection(p, i->rp->p, p.y).y < intersection(p, i->p, p.y).y) {

insertion(i->p, &i);

iftwo = true;

}

insertion(p, &i);

boundu\_event(&i, bev, index5, p.y);

boundu\_event(&i->rp, bev, index5, p.y);

boundd\_event(&i, bev, index5, p.y);

boundd\_event(&i->rp, bev, index5, p.y);

boundr\_event(&i, bev, index5, p.y);

boundr\_event(&i->rp, bev, index5, p.y);

boundl\_event(&i, bev, index5, p.y);

boundl\_event(&i->rp, bev, index5, p.y);

circle\_event(&i->rp, p.y, ev, index3);

circle\_event(&i, p.y, ev, index3);

circle\_event(&i->rp->rp, p.y, ev, index3);

if (iftwo) {

boundu\_event(&i->rp->rp, bev, index5, p.y);

boundd\_event(&i->rp->rp, bev, index5, p.y);

boundl\_event(&i->rp->rp, bev, index5, p.y);

boundr\_event(&i->rp->rp, bev, index5, p.y);

circle\_event(&i->rp->rp->rp, p.y, ev, index3);

}

return;

}

}

insertion(p, b);

boundu\_event(b, bev, index5, p.y);

boundr\_event(b, bev, index5, p.y);

boundl\_event(b, bev, index5, p.y);

boundd\_event(b, bev, index5, p.y);

}

bool ifintersect(point p1, bline p2) {

if (p2->p.y == p1.y) {

return false;

}

double a, b;

if (p2->lp) {

a = intersection(p2->lp->p, p2->p, p1.y).x;

}

if (p2->rp) {

b = intersection(p2->p, p2->rp->p, p1.y).x;

}

if ((!p2->lp || a <= p1.x) && (!p2->rp || p1.x <= b)) {

return true;

}

return false;

}

point intersection(point p1, point p2, double d) {//找斷點

point p;

if (p1.y == p2.y) {

p.x = (p1.x + p2.x) / 2;

}

else if (p2.y == d) {

p.x = p2.x;

}

else if (p1.y == d) {

p.x = p1.x;

p1 = p2;

}

else {

double z0 = 2 \* (p1.y - d);

double z1 = 2 \* (p2.y - d);

double a = 1 / z0 - 1 / z1;

double b = -2 \* (p1.x / z0 - p2.x / z1);

double c = (p1.x \* p1.x + p1.y \* p1.y - d \* d) / z0 - (p2.x \* p2.x + p2.y \* p2.y - d \* d) / z1;

double x1 = (-b - sqrt(b \* b - 4 \* a \* c)) / (2 \* a);

double x2 = (-b + sqrt(b \* b - 4 \* a \* c)) / (2 \* a);

if (p1.x == p2.x) {

if (p1.y < p2.y) {

if (x1 < x2) {

p.x = x1;

}

else {

p.x = x2;

}

}

else {

if (x1 < x2) {

p.x = x2;

}

else {

p.x = x1;

}

}

}

else if (p1.x > p2.x) {

if (p1.y < p2.y) {

if (x1 < x2) {

p.x = x1;

}

else {

p.x = x2;

}

}

else {

if (x1 < x2) {

p.x = x2;

}

else {

p.x = x1;

}

}

}

else {

if (p1.y < p2.y) {

if (x1 < x2) {

p.x = x1;

}

else {

p.x = x2;

}

}

else {

if (x1 < x2) {

p.x = x2;

}

else {

p.x = x1;

}

}

}

}

p.y = (p1.y \* p1.y + (p1.x - p.x) \* (p1.x - p.x) - d \* d) / (2 \* p1.y - 2 \* d);

return p;

}

2. Input Code Format

Three of examples for input use are in below….

\*因字數過大佔篇幅，所以放在檔案中\*

(1)map1.txt

(2)map2.txt

(3)map3.txt

3. Output Code Format

Three of examples for output use are in below….

\*因字數過大佔篇幅，所以放在檔案中\*

(1)o1.txt

(2)o2.txt

(3)o3.txt

**(4) 根據Sub-problem (3)，在該位置安裝一個充電站，則須在原來的Voronoi Diagram外，加一個點變成有*n+1*個點的Voronoi Diagram，設計一程式法建構該新的Voronoi Diagram。**

1. 主程式

#define \_CRT\_SECURE\_NO\_WARNINGS

#include "math.h"

#include "stdio.h"

#include "stdlib.h"

#include "time.h"

int maxx = 0;

int maxy = 0;

struct point {

double x;

double y;

};

struct e;

struct be;

typedef struct bl\* bline;

typedef struct bl {

bline lp, rp;

point p;

e\* ev;

be\* bev;

be\* bevl;

be\* bevr;

be\* bevd;

};

struct e {

double y;

bline b;

point o;

bool valid = true;

};

struct be {

double y;

bline b1, b2;

point o;

bool valid;

};

struct v {

point p;

point p1, p2, p3;

};

struct bv {

point p;

point p1, p2;

};

point ppop(point\* p, int\* index);

point ptop(point\* p);

e\* epop(e\*\* ev, int\* index);

e\* etop(e\*\* ev);

void d(bline b);

void process\_point(point\* p, int\* index, bline\* b, e\*\* ev, int\* index3, be\*\* bev, int\* index5);

void process\_event(e\*\* equeue, int\* index3, v\* v, int\* index4, be\*\* bev, int\* index5, bv\* bvt, int\* index6);

void circle\_event(bline\* b, double y0, e\*\* equeue, int\* index3);

bool circle(point a, point b, point c, double\* y, point\* o);

void insertion(point p, bline\* b);

void insert(point p, bline\* b, e\*\* equeue, int\* index3, be\*\* bequeue, int\* index5);

bool ifintersect(point p1, bline p2);

point intersection(point p1, point p2, double d);

void insertbe(be\*\* bequeue, be\* bev, int\* index5);

be\* bepop(be\*\* bev, int\* index);

be\* betop(be\*\* bev);

bool boundl(point p1, point p2, double\* y, point\* o);

bool boundr(point p1, point p2, double\* y, point\* o);

bool boundu(point p1, point p2, double\* y, point\* o);

bool boundd(point p1, point p2, double\* y, point\* o);

void boundu\_event(bline\* b, be\*\* bequeue, int\* index5, double y0);

void process\_bevent(be\*\* bequeue, int\* index5, bv\* bv, int\* index6);

void boundu\_event(bline\* b, be\*\* bequeue, int\* index5, double y0);

void boundd\_event(bline\* b, be\*\* bequeue, int\* index5, double y0);

void boundr\_event(bline\* b, be\*\* bequeue, int\* index5, double y0);

void boundl\_event(bline\* b, be\*\* bequeue, int\* index5, double y0);

bool boundl(point p1, point p2, double\* y, point\* o);

bool boundr(point p1, point p2, double\* y, point\* o);

bool boundd(point p1, point p2, double\* y, point\* o);

bool boundu(point p1, point p2, double\* y, point\* o);

void inserte(e\*\* equeue, e\* ev, int\* index3);

void print(bv\* bvt, int index6 , v\*, int);

point dt(v\* vt, int \*index4, bv\* bvt, int \*index6, point\* p, int index, point\* p3, int\* index8);

void dv(v\* vt, int i, int\* index4);

void insertv(point p, point\* p2, int\* index8);

void dbv(bv\* bvt, int i, int\* index6);

void checkdbv(point\* p, bv\* bvt, int index9, int\* index6, int index7);

void checkdv(point\* p, v\* vt, int index15, int\* index4, int index7);

void freeb(bline\* b);

int main(void) {

FILE\* map;

char s[10];

printf("請輸入檔案名稱(9位元內)>>");

scanf("%s", s);

map = fopen(s, "r");

int in;

char c;

int index = 0;

int ifeof = 1;

while (ifeof != EOF) {

ifeof = fscanf(map, "%d ", &in);

maxx++;

}

maxx--;

fseek(map, 0, SEEK\_SET);

ifeof = 1;

while (ifeof != EOF) {

ifeof = c = fgetc(map);

if (c == '\n') {

maxy++;

}

}

maxx /= maxy;

fseek(map, 0, SEEK\_SET);

for (int i = 0; i < maxy; i++) {

for (int j = 0; j < maxx; j++) {

fscanf(map, "%d ", &in);

if (in > 0) {

index++;

}

}

}

point\* p = new point[index], \* p2 = new point[index + 1];

index = 0;

int index7 = 0;

fseek(map, 0, SEEK\_SET);

double START, END;

START = clock();

for (int i = 0; i < maxy; i++) {

for (int j = 0; j < maxx; j++) {

fscanf(map, "%d ", &in);

if (in > 0) {

p[index].x = j;

p[index++].y = i;

p2[index7].x = j;

p2[index7++].y = i;

}

}

}

printf("充電站位置：\n");

for (int i = 0; i < index; i++) {

printf("(%.0f,%.0f)\n", p[i].x, p[i].y);

}

int index2 = 0, index4 = 0, index5 = 0, index6 = 0;

int index3 = index \* (index - 1) \* (index - 2) / 6;

bline beachline = NULL;

v\* vt = new v[index3];

bv\* bvt = new bv[index3];

e\*\* ev = new e \* [index3];

be\*\* bev = new be \* [index3];

index3 = 0;

while (index != 0) {

if (index3 != 0 && etop(ev)->y <= ptop(p).y && etop(ev)->y <= betop(bev)->y) {

process\_event(ev, &index3, vt, &index4, bev, &index5, bvt, &index6);

}

else if (index5 == 0 || ptop(p).y <= betop(bev)->y) {

process\_point(p, &index, &beachline, ev, &index3, bev, &index5);

}

else if (index5 != 0) {

process\_bevent(bev, &index5, bvt, &index6);

}

}

while (index3 != 0 || index5 != 0) {

if (index5 == 0 || (index3 != 0 && etop(ev)->y <= betop(bev)->y)) {

process\_event(ev, &index3, vt, &index4, bev, &index5, bvt, &index6);

}

else if (index5 != 0) {

process\_bevent(bev, &index5, bvt, &index6);

}

}

printf("點為：\n");

for (int i = 0; i < index4; i++) {

printf("(%.2f,%.2f)\n", vt[i].p.x, vt[i].p.y);

}

printf("邊界點為：\n");

for (int i = 0; i < index6; i++) {

printf("(%.2f,%.2f)\n", bvt[i].p.x, bvt[i].p.y);

}

freeb(&beachline);

print(bvt, index6, vt, index4);

point\* p3 = new point[index7 + 1];

int index8 = 0;

point op = dt(vt, &index4, bvt, &index6, p2, index7, p3, &index8);

printf("新增點(%.2f,%.2f)\n", op.x, op.y);

insertv(op, p2, &index7);

for (int i = 0; i < index4; i++) {

if (pow(vt[i].p.x - vt[i].p1.x, 2) + pow(vt[i].p.y - vt[i].p1.y, 2) > pow(op.x - vt[i].p.x, 2) + pow(op.y - vt[i].p.y, 2) || pow(vt[i].p.x - vt[i].p2.x, 2) + pow(vt[i].p.y - vt[i].p2.y, 2) > pow(op.x - vt[i].p.x, 2) + pow(op.y - vt[i].p.y, 2) || pow(vt[i].p.x - vt[i].p3.x, 2) + pow(vt[i].p.y - vt[i].p3.y, 2) > pow(op.x - vt[i].p.x, 2) + pow(op.y - vt[i].p.y, 2)) {

insertv(vt[i].p1, p3, &index8);

insertv(vt[i].p2, p3, &index8);

insertv(vt[i].p3, p3, &index8);

dv(vt, i, &index4);

i--;

}

}

for (int i = 0; i < index6; i++) {

if (pow(bvt[i].p.x - bvt[i].p1.x, 2) + pow(bvt[i].p.y - bvt[i].p1.y, 2) > pow(op.x - bvt[i].p.x, 2) + pow(op.y - bvt[i].p.y, 2) || pow(bvt[i].p.x - bvt[i].p2.x, 2) + pow(bvt[i].p.y - bvt[i].p2.y, 2) > pow(op.x - bvt[i].p.x, 2) + pow(op.y - bvt[i].p.y, 2)) {

insertv(bvt[i].p1, p3, &index8);

insertv(bvt[i].p2, p3, &index8);

dbv(bvt, i, &index6);

i--;

}

}

bline b2 = NULL;

int index9 = index6;

int index10 = index4;

while (index8 != 0) {

if (index3 != 0 && etop(ev)->y <= ptop(p3).y && etop(ev)->y <= betop(bev)->y) {

process\_event(ev, &index3, vt, &index4, bev, &index5, bvt, &index6);

}

else if (index5 == 0 || ptop(p3).y <= betop(bev)->y) {

process\_point(p3, &index8, &b2, ev, &index3, bev, &index5);

}

else if (index5 != 0) {

process\_bevent(bev, &index5, bvt, &index6);

}

}

while (index3 != 0 || index5 != 0) {

if (index5 == 0 || (index3 != 0 && etop(ev)->y <= betop(bev)->y)) {

process\_event(ev, &index3, vt, &index4, bev, &index5, bvt, &index6);

}

else if (index5 != 0) {

process\_bevent(bev, &index5, bvt, &index6);

}

}

checkdbv(p2, bvt, index9, &index6, index7);

checkdv(p2, vt, index10, &index4, index7);

freeb(&b2);

delete[]p3;

printf("充電站位置：\n");

for (int i = 0; i < index7; i++) {

printf("(%.2f,%.2f)\n", p2[i].x, p2[i].y);

}

printf("點為：\n");

for (int i = 0; i < index4; i++) {

printf("(%.2f,%.2f)\n", vt[i].p.x, vt[i].p.y);

}

printf("邊界點為：\n");

for (int i = 0; i < index6; i++) {

printf("(%.2f,%.2f)\n", bvt[i].p.x, bvt[i].p.y);

}

print(bvt, index6, vt, index4);

END = clock();

printf("Time:%.0fms", (END - START));

delete[]ev;

delete[]bvt;

delete[]bev;

delete[]vt;

delete[]p;

delete[]p2;

fclose(map);

system("pause");

return 0;

}

void freeb(bline\* b) {

if (\*b == NULL) {

return;

}

freeb(&((\*b)->rp));

free(\*b);

}

point dt(v\* vt, int \*index4, bv\* bvt, int \*index6, point\* p, int index, point\* p3, int \*index8) {

double max = 0;

point o;

int w = 0;

int oindex;

for (int i = 0; i < \*index4; i++) {

if (max <= pow(vt[i].p1.x - vt[i].p.x, 2) + pow(vt[i].p1.y - vt[i].p.y, 2)) {

bool ife2 = true;

for (int j = 0; j < index; j++) {

if (vt[i].p.x == p[j].x && vt[i].p.y == p[j].y) {

ife2 = false;

break;

}

}

if (ife2) {

max = pow(vt[i].p1.x - vt[i].p.x, 2) + pow(vt[i].p1.y - vt[i].p.y, 2);

o = vt[i].p;

oindex = i;

}

}

}

for (int i = 0; i < \*index6; i++) {

if (max <= pow(bvt[i].p1.x - bvt[i].p.x, 2) + pow(bvt[i].p1.y - bvt[i].p.y, 2)) {

bool ife2 = true;

for (int j = 0; j < index; j++) {

if (bvt[i].p.x == p[j].x && bvt[i].p.y == p[j].y) {

ife2 = false;

break;

}

}

if (ife2) {

max = pow(bvt[i].p1.x - bvt[i].p.x, 2) + pow(bvt[i].p1.y - bvt[i].p.y, 2);

o = bvt[i].p;

oindex = i;

w = 1;

}

}

}

double min = maxx \* maxx + maxy \* maxy;

bool ife = true;

for (int i = 0; i < index; i++) {

if (p[i].x == 0 && p[i].y == 0) {

ife = false;

break;

}

}

if (ife) {

for (int i = 0; i < index; i++) {

if (min >= pow(p[i].x, 2) + pow(p[i].y, 2)) {

min = pow(p[i].x, 2) + pow(p[i].y, 2);

}

}

if (max < min) {

max = min;

o.x = 0;

o.y = 0;

w = 2;

}

}

ife = true;

min = maxx \* maxx + maxy \* maxy;

for (int i = 0; i < index; i++) {

if (p[i].x == 0 && p[i].y == maxy) {

ife = false;

break;

}

}

if (ife) {

for (int i = 0; i < index; i++) {

if (min >= pow(p[i].x, 2) + pow(p[i].y - maxy, 2)) {

min = pow(p[i].x, 2) + pow(p[i].y - maxy, 2);

}

}

if (max < min) {

max = min;

o.x = 0;

o.y = maxy;

w = 3;

}

}

ife = true;

min = maxx \* maxx + maxy \* maxy;

for (int i = 0; i < index; i++) {

if (p[i].x == maxx && p[i].y == maxy) {

ife = false;

break;

}

}

if (ife) {

for (int i = 0; i < index; i++) {

if (min >= pow(p[i].x - maxx, 2) + pow(p[i].y - maxy, 2)) {

min = pow(p[i].x - maxx, 2) + pow(p[i].y - maxy, 2);

}

}

if (max <= min) {

max = min;

o.x = maxx;

o.y = maxy;

w = 4;

}

}

ife = true;

min = maxx \* maxx + maxy \* maxy;

for (int i = 0; i < index; i++) {

if (p[i].x == maxx && p[i].y == 0) {

ife = false;

break;

}

}

if (ife) {

for (int i = 0; i < index; i++) {

if (min >= pow(p[i].x - maxx, 2) + pow(p[i].y, 2)) {

min = pow(p[i].x - maxx, 2) + pow(p[i].y, 2);

}

}

if (max < min) {

max = min;

o.x = maxx;

o.y = 0;

w = 5;

}

}

insertv(o, p3, index8);

switch (w) {

case 0:

insertv(vt[oindex].p1, p3, index8);

insertv(vt[oindex].p2, p3, index8);

insertv(vt[oindex].p3, p3, index8);

dv(vt, oindex, index4);

break;

case 1:

insertv(bvt[oindex].p1, p3, index8);

insertv(bvt[oindex].p2, p3, index8);

dbv(bvt, oindex, index6);

break;

case 2:

for (int i = 0; i < index; i++) {

if (max == pow(p[i].x, 2) + pow(p[i].y, 2)) {

insertv(p[i], p3, index8);

}

}

break;

case 3:

for (int i = 0; i < index; i++) {

if (max == pow(p[i].x, 2) + pow(p[i].y - maxy, 2)) {

insertv(p[i], p3, index8);

}

}

break;

case 4:

for (int i = 0; i < index; i++) {

if (max == pow(p[i].x - maxx, 2) + pow(p[i].y - maxy, 2)) {

insertv(p[i], p3, index8);

}

}

break;

case 5:

for (int i = 0; i < index; i++) {

if (max == pow(p[i].x - maxx, 2) + pow(p[i].y, 2)) {

insertv(p[i], p3, index8);

}

}

break;

}

return o;

}

void checkdbv(point\* p, bv\* bvt, int index9, int\* index6, int index7) {

double min;

for (int i = index9; i < \*index6; i++) {

for (int j = 0; j < index9; j++) {

if (bvt[i].p.x == bvt[j].p.x && bvt[i].p.y == bvt[j].p.y) {

dbv(bvt, i--, index6);

break;

}

}

}

for (int i = index9; i < \*index6; i++) {

min = pow(bvt[i].p1.x - bvt[i].p.x, 2) + pow(bvt[i].p1.y - bvt[i].p.y, 2);

for (int j = 0; j < index7; j++) {

if (bvt[i].p2.x != p[j].x || bvt[i].p2.y != p[j].y) {

if (min > pow(p[j].x - bvt[i].p.x, 2) + pow(p[j].y - bvt[i].p.y, 2)) {

dbv(bvt, i--, index6);

break;

}

}

}

}

}

void checkdv(point\* p, v\* vt, int index15, int\* index4, int index7) {

double min;

for (int i = index15; i < \*index4; i++) {

for (int j = 0; j < index15; j++) {

if (vt[i].p.x == vt[j].p.x && vt[i].p.y == vt[j].p.y) {

dv(vt, i--, index4);

break;

}

}

}

for (int i = index15; i < \*index4; i++) {

min = pow(vt[i].p1.x - vt[i].p.x, 2) + pow(vt[i].p1.y - vt[i].p.y, 2);

for (int j = 0; j < index7; j++) {

if ((vt[i].p2.x != p[j].x || vt[i].p2.y != p[j].y) && (vt[i].p3.x != p[j].x || vt[i].p3.y != p[j].y)) {

if (min > pow(p[j].x - vt[i].p.x, 2) + pow(p[j].y - vt[i].p.y, 2)) {

dv(vt, i--, index4);

break;

}

}

}

}

}

void dv(v\* vt, int i, int\* index4) {

for (int j = i; j < \*index4 - 1; j++) {

vt[j] = vt[j + 1];

}

(\*index4)--;

}

void dbv(bv\* bvt, int i, int\* index6) {

for (int j = i; j < \*index6 - 1; j++) {

bvt[j] = bvt[j + 1];

}

(\*index6)--;

}

void insertv(point p, point\* p2, int\* index8) {

if (\*index8 == 0) {

p2[(\*index8)++] = p;

return;

}

bool ife = true;

for (int i = 0; i < \*index8; i++) {

if (p2[i].x == p.x && p2[i].y == p.y) {

ife = false;

break;

}

}

if (ife) {

if (p2[\*index8 - 1].y < p.y || (p2[\*index8 - 1].y == p.y && p2[\*index8 - 1].x <= p.x)) {

p2[(\*index8)++] = p;

}

else if (p2[0].y > p.y || (p2[0].x >= p.x && p2[0].y == p.y)) {

for (int i = \*index8; i > 0; i--) {

p2[i] = p2[i - 1];

}

p2[0] = p;

(\*index8)++;

}

else {

int x;

for (int i = 0; i < \*index8 - 1; i++) {

if (p2[i].y < p.y && p2[i + 1].y > p.y) {

x = i + 1;

break;

}

else if (p2[i].y == p.y && p2[i].x <= p.x && p2[i + 1].y == p.y && p2[i + 1].x >= p.x) {

x = i + 1;

break;

}

else if (p2[i].y == p.y && p2[i].x <= p.x && p2[i + 1].y > p.y) {

x = i + 1;

break;

}

else if (p2[i].y < p.y&& p2[i + 1].x >= p.x && p2[i + 1].y == p.y) {

x = i + 1;

break;

}

}

for (int i = \*index8; i > x; i--) {

p2[i] = p2[i - 1];

}

p2[x] = p;

(\*index8)++;

}

}

}

void print(bv\* bvt, int index6, v\* vt, int index4) {

bool ife = true;

for (int i = 0; i < index6; i++) {

if (bvt[i].p.x == 0 && bvt[i].p.y == 0) {

ife = false;

}

}

for (int i = 0; i < index4; i++) {

if (vt[i].p.x == 0 && vt[i].p.y == 0) {

ife = false;

}

}

if (ife) {

printf("(0.00,0.00)\n");

}

ife = true;

for (int i = 0; i < index6; i++) {

if (bvt[i].p.x == 0 && bvt[i].p.y == maxy) {

ife = false;

}

}

for (int i = 0; i < index4; i++) {

if (vt[i].p.x == 0 && vt[i].p.y == maxy) {

ife = false;

}

}

if (ife) {

printf("(0.00,%d.00)\n", maxy);

}

ife = true;

for (int i = 0; i < index6; i++) {

if (bvt[i].p.x == maxx && bvt[i].p.y == 0) {

ife = false;

}

}

for (int i = 0; i < index4; i++) {

if (vt[i].p.x == maxx && vt[i].p.y == 0) {

ife = false;

}

}

if (ife) {

printf("(%d.00,0.00)\n", maxx);

}

ife = true;

for (int i = 0; i < index6; i++) {

if (bvt[i].p.x == maxx && bvt[i].p.y == maxy) {

ife = false;

}

}

for (int i = 0; i < index4; i++) {

if (vt[i].p.x == maxx && vt[i].p.y == maxy) {

ife = false;

}

}

if (ife) {

printf("(%d.00,%d.00)\n", maxx, maxy);

}

}

point ppop(point\* p, int\* index) {

point p0 = p[0];

for (int i = 0; i < \*index - 1; i++) {

p[i] = p[i + 1];

}

(\*index)--;

return p0;

}

point ptop(point\* p) {

return p[0];

}

e\* epop(e\*\* ev, int\* index) {

e\* e0 = ev[0];

for (int i = 0; i < \*index - 1; i++) {

ev[i] = ev[i + 1];

}

(\*index)--;

return e0;

}

e\* etop(e\*\* ev) {

return ev[0];

}

be\* bepop(be\*\* bev, int\* index) {

be\* be0 = bev[0];

for (int i = 0; i < \*index - 1; i++) {

bev[i] = bev[i + 1];

}

(\*index)--;

return be0;

}

be\* betop(be\*\* bev) {

return bev[0];

}

void d(bline b) {

if (b->bev) {

b->bev->valid = false;

b->bev = NULL;

}

if (b->lp->bev) {

b->lp->bev->valid = false;

b->lp->bev = NULL;

}

if (b->bevd) {

b->bevd->valid = false;

b->bevd = NULL;

}

if (b->lp->bevd) {

b->lp->bevd->valid = false;

b->lp->bevd = NULL;

}

if (b->bevr) {

b->bevr->valid = false;

b->bevr = NULL;

}

if (b->lp->bevr) {

b->lp->bevr->valid = false;

b->lp->bevr = NULL;

}

if (b->bevl) {

b->bevl->valid = false;

b->bevl = NULL;

}

if (b->lp->bevl) {

b->lp->bevl->valid = false;

b->lp->bevl = NULL;

}

if (b->ev) {

b->ev->valid = false;

b->ev = NULL;

}

if (b->lp->ev) {

b->lp->ev->valid = false;

b->lp->ev = NULL;

}

if (b->rp->ev) {

b->rp->ev->valid = false;

b->rp->ev = NULL;

}

if (b->rp && b->lp) {

b->rp->lp = b->lp;

b->lp->rp = b->rp;

}

else if (b->rp) {

b->rp->lp = NULL;

}

else if (b->lp) {

b->lp->rp = NULL;

}

}

void process\_point(point\* p, int\* index, bline\* b, e\*\* ev, int\* index3, be\*\* bev, int\* index5) {

point p0 = ppop(p, index);

insert(p0, b, ev, index3, bev, index5);

}

void process\_event(e\*\* equeue, int\* index3, v\* v, int\* index4, be\*\* bequeue, int\* index5, bv\* bvt, int\* index6) {

e\* ev = epop(equeue, index3);

if (ev->valid) {

if (ev->o.x > maxx || ev->o.x < 0 || ev->o.y > maxy || ev->o.y < 0) {

}

else {

v[\*index4].p = ev->o;

v[\*index4].p2 = ev->b->p;

v[\*index4].p1 = ev->b->lp->p;

v[(\*index4)++].p3 = ev->b->rp->p;

}

d(ev->b);

boundu\_event(&ev->b->lp, bequeue, index5, ev->y);

boundd\_event(&ev->b->lp, bequeue, index5, ev->y);

boundr\_event(&ev->b->lp, bequeue, index5, ev->y);

boundl\_event(&ev->b->lp, bequeue, index5, ev->y);

circle\_event(&ev->b->lp, ev->y, equeue, index3);

circle\_event(&ev->b->rp, ev->y, equeue, index3);

if (ev->b->lp) {

free(ev->b);

}

else if (!ev->b->lp && ev->b->rp) {

ev->b = ev->b->rp;

free(ev->b->lp);

ev->b->lp = NULL;

}

else {

ev->b = NULL;

}

}

free(ev);

}

void process\_bevent(be\*\* bequeue, int\* index5, bv\* bv, int\* index6) {

be\* bev = bepop(bequeue, index5);

if (bev->valid) {

bv[\*index6].p = bev->o;

bv[\*index6].p2 = bev->b1->p;

bv[(\*index6)++].p1 = bev->b2->p;

if (bev->o.y == 0) {

bev->b1->bev = NULL;

}

else if (bev->o.y == maxy) {

bev->b1->bevd = NULL;

}

else if (bev->o.x == 0) {

bev->b1->bevl = NULL;

}

else if (bev->o.x == maxx) {

bev->b1->bevr = NULL;

}

}

free(bev);

}

void boundu\_event(bline\* b, be\*\* bequeue, int\* index5, double y0) {

if (!\*b || !(\*b)->rp) {

return;

}

double y;

point o;

be\* bev;

if (boundu((\*b)->p, (\*b)->rp->p, &y, &o) && y >= y0) {

if (o.x <= (\*b)->p.x) {

if (!(\*b)->lp || intersection((\*b)->lp->p, (\*b)->rp->p, y).y < o.y) {

}

else {

return;

}

}

else if (o.x >= (\*b)->rp->p.x) {

if (!(\*b)->rp->rp || intersection((\*b)->p, (\*b)->rp->rp->p, y).y < o.y) {

}

else {

return;

}

}

bev = (be\*)malloc(sizeof(\*bev));

bev->b1 = \*b;

bev->b2 = (\*b)->rp;

bev->o = o;

bev->valid = true;

bev->y = y;

(\*b)->bev = bev;

insertbe(bequeue, bev, index5);

}

}

void boundd\_event(bline\* b, be\*\* bequeue, int\* index5, double y0) {

if (!\*b || !(\*b)->rp) {

return;

}

double y;

point o;

be\* bev = NULL;

if (boundd((\*b)->p, (\*b)->rp->p, &y, &o) && y >= y0) {

bev = (be\*)malloc(sizeof(\*bev));

bev->b1 = \*b;

bev->b2 = (\*b)->rp;

bev->o = o;

bev->valid = true;

bev->y = y;

(\*b)->bevd = bev;

insertbe(bequeue, bev, index5);

}

}

void boundr\_event(bline\* b, be\*\* bequeue, int\* index5, double y0) {

if (!\*b || !(\*b)->rp) {

return;

}

double y;

point o;

be\* bev;

if (boundr((\*b)->p, (\*b)->rp->p, &y, &o) && y >= y0) {

bev = (be\*)malloc(sizeof(\*bev));

bev->b1 = \*b;

bev->b2 = (\*b)->rp;

bev->o = o;

bev->valid = true;

bev->y = y;

(\*b)->bevr = bev;

insertbe(bequeue, bev, index5);

}

}

void boundl\_event(bline\* b, be\*\* bequeue, int\* index5, double y0) {

if (!\*b || !(\*b)->rp) {

return;

}

double y;

point o;

be\* bev;

if (boundl((\*b)->p, (\*b)->rp->p, &y, &o) && y >= y0) {

bev = (be\*)malloc(sizeof(\*bev));

bev->b1 = \*b;

bev->b2 = (\*b)->rp;

bev->o = o;

bev->valid = true;

bev->y = y;

(\*b)->bevl = bev;

insertbe(bequeue, bev, index5);

}

}

bool boundl(point p1, point p2, double\* y, point\* o) {

if (p1.y >= p2.y) {

return false;

}

double oy = ((p1.x \* p1.x + p1.y \* p1.y) - (p2.x \* p2.x + p2.y \* p2.y)) / (2 \* (p1.y - p2.y));

if (oy < maxy && oy > 0) {

double y0 = (2 \* oy + sqrt(4 \* oy \* oy - 4 \* (2 \* oy \* p1.y - p1.x \* p1.x - p1.y \* p1.y))) / 2;

o->x = 0;

o->y = oy;

\*y = y0;

return true;

}

return false;

}

bool boundr(point p1, point p2, double\* y, point\* o) {

if (p1.y <= p2.y) {

return false;

}

double oy = (2 \* maxx \* (p2.x - p1.x) + p1.x \* p1.x + p1.y \* p1.y - p2.x \* p2.x - p2.y \* p2.y) / (2 \* (p1.y - p2.y));

if (oy < maxy && oy > 0) {

double y0 = (2 \* oy + sqrt(4 \* oy \* oy - 4 \* (2 \* oy \* p1.y + 2 \* p1.x \* maxx - maxx \* maxx - p1.x \* p1.x - p1.y \* p1.y))) / 2;

o->x = maxx;

o->y = oy;

\*y = y0;

return true;

}

return false;

}

bool boundd(point p1, point p2, double\* y, point\* o) {

if (p1.x == p2.x) {

return false;

}

double ox = (2 \* maxy \* (p2.y - p1.y) + p1.x \* p1.x + p1.y \* p1.y - p2.x \* p2.x - p2.y \* p2.y) / (2 \* (p1.x - p2.x));

if (ox <= maxy && ox >= 0) {

double y0 = (2 \* maxy + sqrt(4 \* maxy \* maxy - 4 \* (2 \* ox \* p1.x + 2 \* p1.y \* maxy - ox \* ox - p1.x \* p1.x - p1.y \* p1.y))) / 2;

o->x = ox;

o->y = maxy;

\*y = y0;

return true;

}

return false;

}

bool boundu(point p1, point p2, double\* y, point\* o) {

if (p1.x == p2.x) {

return false;

}

double ox = ((p1.x \* p1.x + p1.y \* p1.y) - (p2.x \* p2.x + p2.y \* p2.y)) / (2 \* (p1.x - p2.x));

if (ox <= maxx && ox >= 0) {

double y0 = sqrt(ox \* ox - 2 \* ox \* p1.x + p1.x \* p1.x + p1.y \* p1.y);

o->x = ox;

o->y = 0;

\*y = y0;

return true;

}

return false;

}

void inserte(e\*\* equeue, e\* ev, int\* index3) {

if (\*index3 == 0) {

equeue[0] = ev;

(\*index3)++;

return;

}

if (equeue[0]->y >= ev->y) {

for (int j = \*index3; j > 0; j--) {

equeue[j] = equeue[j - 1];

}

equeue[0] = ev;

(\*index3)++;

return;

}

if (equeue[\*index3 - 1]->y <= ev->y) {

equeue[\*index3] = ev;

(\*index3)++;

return;

}

for (int i = 0; i < \*index3 - 1; i++) {

if (equeue[i]->y <= ev->y && equeue[i + 1]->y >= ev->y) {

for (int j = \*index3; j > i + 1; j--) {

equeue[j] = equeue[j - 1];

}

equeue[i + 1] = ev;

(\*index3)++;

break;

}

}

}

void insertbe(be\*\* bequeue, be\* bev, int\* index5) {

if (\*index5 == 0) {

bequeue[0] = bev;

(\*index5)++;

return;

}

if (bequeue[0]->y >= bev->y) {

for (int j = \*index5; j > 0; j--) {

bequeue[j] = bequeue[j - 1];

}

bequeue[0] = bev;

(\*index5)++;

return;

}

if (bequeue[\*index5 - 1]->y <= bev->y) {

bequeue[\*index5] = bev;

(\*index5)++;

return;

}

for (int i = 0; i < \*index5 - 1; i++) {

if (bequeue[i]->y <= bev->y && bequeue[i + 1]->y >= bev->y) {

for (int j = \*index5; j > i + 1; j--) {

bequeue[j] = bequeue[j - 1];

}

bequeue[i + 1] = bev;

(\*index5)++;

break;

}

}

}

void circle\_event(bline\* b, double y0, e\*\* equeue, int\* index3) {

if (!\*b || !(\*b)->lp || !(\*b)->rp) {

return;

}

double y;

point o;

e\* ev;

if (circle((\*b)->lp->p, (\*b)->p, (\*b)->rp->p, &y, &o) && y >= y0) {

ev = (e\*)malloc(sizeof(\*ev));

ev->valid = true;

ev->y = y;

ev->o = o;

ev->b = \*b;

(\*b)->ev = ev;

inserte(equeue, ev, index3);

}

}

bool circle(point a, point b, point c, double\* y, point\* o) {

if ((b.y - a.y) \* (c.x - a.x) - (c.y - a.y) \* (b.x - a.x) > 0) {

return false;

}

double A = 2 \* (b.x - a.x), B = 2 \* (b.y - a.y),

A2 = 2 \* (c.x - b.x), B2 = 2 \* (c.y - b.y),

C = b.x \* b.x + b.y \* b.y - a.x \* a.x - a.y \* a.y,

C2 = c.x \* c.x + c.y \* c.y - b.x \* b.x - b.y \* b.y,

D = (A \* B2 - A2 \* B);

if (D == 0) {

return false;

}

o->x = (B2 \* C - B \* C2) / D;

o->y = (A \* C2 - A2 \* C) / D;

\*y = o->y + sqrt(pow(a.y - o->y, 2) + pow(a.x - o->x, 2));

return true;

}

void insertion(point p, bline\* b) {

if ((\*b)->bev) {

(\*b)->bev->valid = false;

(\*b)->bev = NULL;

}

if ((\*b)->bevd) {

(\*b)->bevd->valid = false;

(\*b)->bevd = NULL;

}

if ((\*b)->bevr) {

(\*b)->bevr->valid = false;

(\*b)->bevr = NULL;

}

if ((\*b)->bevl) {

(\*b)->bevl->valid = false;

(\*b)->bevl = NULL;

}

if ((\*b)->ev) {

(\*b)->ev->valid = false;

(\*b)->ev = NULL;

}

if ((\*b)->rp && (\*b)->rp->ev) {

(\*b)->rp->ev->valid = false;

(\*b)->rp->ev = NULL;

}

bline i, j = \*b;

i = (bline)malloc(sizeof(\*i));

i->p = p;

i->lp = i->rp = NULL;

i->ev = NULL;

i->bev = NULL;

i->bevd = NULL;

i->bevr = NULL;

i->bevl = NULL;

i->lp = j;

if ((\*b)->rp) {

i->rp = j->rp;

j->rp->lp = i;

}

j->rp = i;

}

void insert(point p, bline\* b, e\*\* ev, int\* index3, be\*\* bev, int\* index5) {

if (!\*b) {

bline a = NULL;

a = (bline)malloc(sizeof(\*a));

a->p = p;

a->lp = a->rp = NULL;

a->ev = NULL;

a->bev = NULL;

a->bevd = NULL;

a->bevr = NULL;

a->bevl = NULL;

\*b = a;

return;

}

bool iftwo = false;

int j = 0;

for (bline i = \*b; i; i = i->rp, j++) {

if (ifintersect(p, i)) {

if (j == 0) {

if (intersection(p, i->p, p.y).y > 0) {

bline a;

a = (bline)malloc(sizeof(\*a));

a->p = i->p;

a->lp = a->rp = NULL;

a->ev = NULL;

a->bev = NULL;

a->bevd = NULL;

a->bevr = NULL;

a->bevl = NULL;

a->rp = i;

i->lp = a;

insertion(p, &i->lp);

boundu\_event(&i->lp, bev, index5, p.y);

boundu\_event(&i->lp->lp, bev, index5, p.y);

boundr\_event(&i->lp, bev, index5, p.y);

boundr\_event(&i->lp->lp, bev, index5, p.y);

boundl\_event(&i->lp, bev, index5, p.y);

boundl\_event(&i->lp->lp, bev, index5, p.y);

boundd\_event(&i->lp, bev, index5, p.y);

boundd\_event(&i->lp->lp, bev, index5, p.y);

circle\_event(&i, p.y, ev, index3);

circle\_event(&i->lp, p.y, ev, index3);

\*b = i->lp->lp;

}

else if (intersection(p, i->p, p.y).x > i->p.x) {

insertion(p, &i);

boundu\_event(&i, bev, index5, p.y);

boundu\_event(&i->rp, bev, index5, p.y);

boundd\_event(&i, bev, index5, p.y);

boundd\_event(&i->rp, bev, index5, p.y);

boundr\_event(&i, bev, index5, p.y);

boundr\_event(&i->rp, bev, index5, p.y);

boundl\_event(&i, bev, index5, p.y);

boundl\_event(&i->rp, bev, index5, p.y);

circle\_event(&i->rp, p.y, ev, index3);

circle\_event(&i->rp->rp, p.y, ev, index3);

}

else {

bline a;

a = (bline)malloc(sizeof(\*a));

a->p = p;

a->lp = a->rp = NULL;

a->ev = NULL;

a->bev = NULL;

a->bevd = NULL;

a->bevr = NULL;

a->bevl = NULL;

a->rp = i;

i->lp = a;

boundu\_event(&i->lp, bev, index5, p.y);

boundd\_event(&i->lp, bev, index5, p.y);

boundr\_event(&i->lp, bev, index5, p.y);

boundl\_event(&i->lp, bev, index5, p.y);

circle\_event(&i, p.y, ev, index3);

\*b = i->lp;

}

return;

}

if (!i->rp) {

if (intersection(i->p, p, p.y).y > 0) {

insertion(i->p, &i);

insertion(p, &i);

boundu\_event(&i, bev, index5, p.y);

boundu\_event(&i->rp, bev, index5, p.y);

boundd\_event(&i, bev, index5, p.y);

boundd\_event(&i->rp, bev, index5, p.y);

boundr\_event(&i, bev, index5, p.y);

boundr\_event(&i->rp, bev, index5, p.y);

boundl\_event(&i, bev, index5, p.y);

boundl\_event(&i->rp, bev, index5, p.y);

circle\_event(&i, p.y, ev, index3);

circle\_event(&i->rp, p.y, ev, index3);

}

else if (intersection(p, i->p, p.y).x > i->p.x) {

insertion(p, &i);

boundu\_event(&i, bev, index5, p.y);

boundd\_event(&i, bev, index5, p.y);

boundr\_event(&i, bev, index5, p.y);

boundl\_event(&i, bev, index5, p.y);

circle\_event(&i, p.y, ev, index3);

}

else {

insertion(p, &i->lp);

boundu\_event(&i->lp, bev, index5, p.y);

boundu\_event(&i->lp->lp, bev, index5, p.y);

boundr\_event(&i->lp, bev, index5, p.y);

boundr\_event(&i->lp->lp, bev, index5, p.y);

boundd\_event(&i->lp, bev, index5, p.y);

boundd\_event(&i->lp->lp, bev, index5, p.y);

boundl\_event(&i->lp, bev, index5, p.y);

boundl\_event(&i->lp->lp, bev, index5, p.y);

circle\_event(&i->lp, p.y, ev, index3);

circle\_event(&i->lp->lp, p.y, ev, index3);

}

return;

}

if (i->rp && intersection(p, i->rp->p, p.y).y < intersection(p, i->p, p.y).y) {

insertion(i->p, &i);

iftwo = true;

}

insertion(p, &i);

boundu\_event(&i, bev, index5, p.y);

boundu\_event(&i->rp, bev, index5, p.y);

boundd\_event(&i, bev, index5, p.y);

boundd\_event(&i->rp, bev, index5, p.y);

boundr\_event(&i, bev, index5, p.y);

boundr\_event(&i->rp, bev, index5, p.y);

boundl\_event(&i, bev, index5, p.y);

boundl\_event(&i->rp, bev, index5, p.y);

circle\_event(&i->rp, p.y, ev, index3);

circle\_event(&i, p.y, ev, index3);

circle\_event(&i->rp->rp, p.y, ev, index3);

if (iftwo) {

boundu\_event(&i->rp->rp, bev, index5, p.y);

boundd\_event(&i->rp->rp, bev, index5, p.y);

boundl\_event(&i->rp->rp, bev, index5, p.y);

boundr\_event(&i->rp->rp, bev, index5, p.y);

circle\_event(&i->rp->rp->rp, p.y, ev, index3);

}

return;

}

}

insertion(p, b);

boundu\_event(b, bev, index5, p.y);

boundr\_event(b, bev, index5, p.y);

boundl\_event(b, bev, index5, p.y);

boundd\_event(b, bev, index5, p.y);

}

bool ifintersect(point p1, bline p2) {

if (p2->p.y == p1.y) {

return false;

}

double a, b;

if (p2->lp) {

a = intersection(p2->lp->p, p2->p, p1.y).x;

}

if (p2->rp) {

b = intersection(p2->p, p2->rp->p, p1.y).x;

}

if ((!p2->lp || a <= p1.x) && (!p2->rp || p1.x <= b)) {

return true;

}

return false;

}

point intersection(point p1, point p2, double d) {//找斷點

point p;

if (p1.y == p2.y) {

p.x = (p1.x + p2.x) / 2;

}

else if (p2.y == d) {

p.x = p2.x;

}

else if (p1.y == d) {

p.x = p1.x;

p1 = p2;

}

else {

double z0 = 2 \* (p1.y - d);

double z1 = 2 \* (p2.y - d);

double a = 1 / z0 - 1 / z1;

double b = -2 \* (p1.x / z0 - p2.x / z1);

double c = (p1.x \* p1.x + p1.y \* p1.y - d \* d) / z0 - (p2.x \* p2.x + p2.y \* p2.y - d \* d) / z1;

double x1 = (-b - sqrt(b \* b - 4 \* a \* c)) / (2 \* a);

double x2 = (-b + sqrt(b \* b - 4 \* a \* c)) / (2 \* a);

if (p1.x == p2.x) {

if (p1.y < p2.y) {

if (x1 < x2) {

p.x = x1;

}

else {

p.x = x2;

}

}

else {

if (x1 < x2) {

p.x = x2;

}

else {

p.x = x1;

}

}

}

else if (p1.x > p2.x) {

if (p1.y < p2.y) {

if (x1 < x2) {

p.x = x1;

}

else {

p.x = x2;

}

}

else {

if (x1 < x2) {

p.x = x2;

}

else {

p.x = x1;

}

}

}

else {

if (p1.y < p2.y) {

if (x1 < x2) {

p.x = x1;

}

else {

p.x = x2;

}

}

else {

if (x1 < x2) {

p.x = x2;

}

else {

p.x = x1;

}

}

}

}

p.y = (p1.y \* p1.y + (p1.x - p.x) \* (p1.x - p.x) - d \* d) / (2 \* p1.y - 2 \* d);

return p;

}

2. Input Code Format

Three of examples for input use are in below….

\*因字數過大佔篇幅，所以放在檔案中\*

(1)map1.txt

(2)map2.txt

(3)map3.txt

3. Output Code Format

Three of examples for output use are in below….

\*因字數過大佔篇幅，所以放在檔案中\*

(1)o1.txt

(2)o2.txt

(3)o3.txt

**(5) 重複Sub-problems (3) and (4)直到最遠距離滿足一個門檻值(Threshold value *k*, 如500m，*k*值可自行設定)，並輸出總共需要幾個充電站、目前離充電站最遠的距離與*k*值。**

1. 主程式

#define \_CRT\_SECURE\_NO\_WARNINGS

#include "math.h"

#include "stdio.h"

#include "stdlib.h"

#include "time.h"

int maxx = 0;

int maxy = 0;

struct point {

double x;

double y;

};

struct e;

struct be;

typedef struct bl\* bline;

typedef struct bl {

bline lp, rp;

point p;

e\* ev;

be\* bev;

be\* bevl;

be\* bevr;

be\* bevd;

};

struct e {

double y;

bline b;

point o;

bool valid = true;

};

struct be {

double y;

bline b1, b2;

point o;

bool valid;

};

struct v {

point p;

point p1, p2, p3;

};

struct bv {

point p;

point p1, p2;

};

point ppop(point\* p, int\* index);

point ptop(point\* p);

e\* epop(e\*\* ev, int\* index);

e\* etop(e\*\* ev);

void d(bline b);

void process\_point(point\* p, int\* index, bline\* b, e\*\* ev, int\* index3, be\*\* bev, int\* index5);

void process\_event(e\*\* equeue, int\* index3, v\* v, int\* index4, be\*\* bev, int\* index5, bv\* bvt, int\* index6);

void circle\_event(bline\* b, double y0, e\*\* equeue, int\* index3);

bool circle(point a, point b, point c, double\* y, point\* o);

void insertion(point p, bline\* b);

void insert(point p, bline\* b, e\*\* equeue, int\* index3, be\*\* bequeue, int\* index5);

bool ifintersect(point p1, bline p2);

point intersection(point p1, point p2, double d);

void insertbe(be\*\* bequeue, be\* bev, int\* index5);

be\* bepop(be\*\* bev, int\* index);

be\* betop(be\*\* bev);

bool boundl(point p1, point p2, double\* y, point\* o);

bool boundr(point p1, point p2, double\* y, point\* o);

bool boundu(point p1, point p2, double\* y, point\* o);

bool boundd(point p1, point p2, double\* y, point\* o);

void boundu\_event(bline\* b, be\*\* bequeue, int\* index5, double y0);

void process\_bevent(be\*\* bequeue, int\* index5, bv\* bv, int\* index6);

void boundu\_event(bline\* b, be\*\* bequeue, int\* index5, double y0);

void boundd\_event(bline\* b, be\*\* bequeue, int\* index5, double y0);

void boundr\_event(bline\* b, be\*\* bequeue, int\* index5, double y0);

void boundl\_event(bline\* b, be\*\* bequeue, int\* index5, double y0);

bool boundl(point p1, point p2, double\* y, point\* o);

bool boundr(point p1, point p2, double\* y, point\* o);

bool boundd(point p1, point p2, double\* y, point\* o);

bool boundu(point p1, point p2, double\* y, point\* o);

void inserte(e\*\* equeue, e\* ev, int\* index3);

void print(bv\* bvt, int index6, v\* vt, int index4);

point dt(v\* vt, int\* index4, bv\* bvt, int\* index6, point\* p, int index, point\* p3, int\* index8);

void dv(v\* vt, int i, int\* index4);

void insertv(point p, point\* p2, int\* index8);

double dtlong(v\* vt, int index4, bv\* bvt, int index6, point\* p, int index);

void dbv(bv\* bvt, int i, int\* index6);

void checkdbv(point\* p, bv\* bvt, int index9, int\* index6, int index7);

void checkdv(point\* p, v\* vt, int index15, int\* index4, int index7);

void freeb(bline\* b);

int main(void) {

FILE\* map;

char s[10];

printf("請輸入檔案名稱(9位元內)>>");

scanf("%s", s);

map = fopen(s, "r");

int in;

char c;

int index = 0;

int ifeof = 1;

while (ifeof != EOF) {

ifeof = fscanf(map, "%d ", &in);

maxx++;

}

maxx--;

fseek(map, 0, SEEK\_SET);

ifeof = 1;

while (ifeof != EOF) {

ifeof = c = fgetc(map);

if (c == '\n') {

maxy++;

}

}

maxx /= maxy;

fseek(map, 0, SEEK\_SET);

for (int i = 0; i < maxy; i++) {

for (int j = 0; j < maxx; j++) {

fscanf(map, "%d ", &in);

if (in > 0) {

index++;

}

}

}

point\* p = new point[index], \* p2 = new point[index];

index = 0;

int index7 = 0;

fseek(map, 0, SEEK\_SET);

for (int i = 0; i < maxy; i++) {

for (int j = 0; j < maxx; j++) {

fscanf(map, "%d ", &in);

if (in > 0) {

p[index].x = j;

p[index++].y = i;

p2[index7].x = j;

p2[index7++].y = i;

}

}

}

printf("充電站位置：\n");

for (int i = 0; i < index; i++) {

printf("(%.0f,%.0f)\n", p[i].x, p[i].y);

}

int index2 = 0, index4 = 0, index5 = 0, index6 = 0;

int index3 = index \* (index - 1) \* (index - 2) / 6;

bline beachline = NULL;

v\* vt = new v[index3];

bv\* bvt = new bv[index3];

e\*\* ev = new e \* [index3];

be\*\* bev = new be \* [index3];

index3 = 0;

while (index != 0) {

if (index3 != 0 && etop(ev)->y <= ptop(p).y && etop(ev)->y <= betop(bev)->y) {

process\_event(ev, &index3, vt, &index4, bev, &index5, bvt, &index6);

}

else if (index5 == 0 || ptop(p).y <= betop(bev)->y) {

process\_point(p, &index, &beachline, ev, &index3, bev, &index5);

}

else if (index5 != 0) {

process\_bevent(bev, &index5, bvt, &index6);

}

}

while (index3 != 0 || index5 != 0) {

if (index5 == 0 || (index3 != 0 && etop(ev)->y <= betop(bev)->y)) {

process\_event(ev, &index3, vt, &index4, bev, &index5, bvt, &index6);

}

else if (index5 != 0) {

process\_bevent(bev, &index5, bvt, &index6);

}

}

printf("點為：\n");

for (int i = 0; i < index4; i++) {

printf("(%.2f,%.2f)\n", vt[i].p.x, vt[i].p.y);

}

printf("邊界點為：\n");

for (int i = 0; i < index6; i++) {

printf("(%.2f,%.2f)\n", bvt[i].p.x, bvt[i].p.y);

}

freeb(&beachline);

print(bvt, index6, vt, index4);

double k0;

printf("請輸入最遠距離門檻值(公里)\*注意：門檻值太低，數值很可能會overflow或是空間不足計算生產點，如果跑不出來，請將門檻值調大!>>");

scanf("%lf", &k0);

int n = 0;

int number = index7 + round(maxx \* maxy / (k0 + 1) / (k0 + 1) + 1);

int maxxy = 0;

if (maxx > maxy) {

maxxy = maxx;

}

else {

maxxy = maxy;

}

double START, END;

START = clock();

point\* p4 = (point\*)malloc(number\*sizeof(\*p4));

int index10 = 0;

for (int i = 0; i < index7; i++) {

p4[index10++] = p2[i];

}

int number2 = number \* (number - 1) \* (number - 2) / 6;

if (number2 < index6) {

number2 = index6;

}

if (number2 < index4) {

number2 = index4;

}

double k = dtlong(vt, index4, bvt, index6, p4, index10);

int index11 = 0, index12 = 0, index13 = 0, index14 = 0;

v\* vt2 = (v\*)malloc(number2 \* sizeof(\*vt2));

bv\* bvt2 = (bv\*)malloc(number2 \* sizeof(\*bvt2));

for (int i = 0; i < index4; i++) {

vt2[index11++] = vt[i];

}

for (int i = 0; i < index6; i++) {

bvt2[index12++] = bvt[i];

}

while (k > k0) {

point\* p3 = (point\*)malloc(number \* sizeof(\*p3));

int index8 = 0;

point op = dt(vt2, &index11, bvt2, &index12, p4, index10, p3, &index8);

e\*\* ev2 = (e\*\*)malloc(number2 \* sizeof(\*ev2));

be\*\* bev2 = (be\*\*)malloc(number2 \* sizeof(\*bev2));

printf("新增點(%.2f,%.2f)\n", op.x, op.y);

insertv(op, p4, &index10);

for (int i = 0; i < index11; i++) {

if (pow(vt2[i].p.x - vt2[i].p1.x, 2) + pow(vt2[i].p.y - vt2[i].p1.y, 2) > pow(op.x - vt2[i].p.x, 2) + pow(op.y - vt2[i].p.y, 2) || pow(vt2[i].p.x - vt2[i].p2.x, 2) + pow(vt2[i].p.y - vt2[i].p2.y, 2) > pow(op.x - vt2[i].p.x, 2) + pow(op.y - vt2[i].p.y, 2) || pow(vt2[i].p.x - vt2[i].p3.x, 2) + pow(vt2[i].p.y - vt2[i].p3.y, 2) > pow(op.x - vt2[i].p.x, 2) + pow(op.y - vt2[i].p.y, 2)) {

insertv(vt2[i].p1, p3, &index8);

insertv(vt2[i].p2, p3, &index8);

insertv(vt2[i].p3, p3, &index8);

dv(vt2, i, &index11);

i--;

}

}

for (int i = 0; i < index12; i++) {

if (pow(bvt2[i].p.x - bvt2[i].p1.x, 2) + pow(bvt2[i].p.y - bvt2[i].p1.y, 2) > pow(op.x - bvt2[i].p.x, 2) + pow(op.y - bvt2[i].p.y, 2) || pow(bvt2[i].p.x - bvt2[i].p2.x, 2) + pow(bvt2[i].p.y - bvt2[i].p2.y, 2) > pow(op.x - bvt2[i].p.x, 2) + pow(op.y - bvt2[i].p.y, 2)) {

insertv(bvt2[i].p1, p3, &index8);

insertv(bvt2[i].p2, p3, &index8);

dbv(bvt2, i, &index12);

i--;

}

}

bline b2 = NULL;

int index9 = index12;

int index15 = index11;

while (index8 != 0) {

if (index13 != 0 && etop(ev2)->y <= ptop(p3).y && etop(ev2)->y <= betop(bev2)->y) {

process\_event(ev2, &index13, vt2, &index11, bev2, &index14, bvt2, &index12);

}

else if (index14 == 0 || ptop(p3).y <= betop(bev2)->y) {

process\_point(p3, &index8, &b2, ev2, &index13, bev2, &index14);

}

else if (index14 != 0) {

process\_bevent(bev2, &index14, bvt2, &index12);

}

}

while (index13 != 0 || index14 != 0) {

if (index14 == 0 || (index13 != 0 && etop(ev2)->y <= betop(bev2)->y)) {

process\_event(ev2, &index13, vt2, &index11, bev2, &index14, bvt2, &index12);

}

else if (index14 != 0) {

process\_bevent(bev2, &index14, bvt2, &index12);

}

}

checkdbv(p4, bvt2, index9, &index12, index10);

checkdv(p4, vt2, index15, &index11, index10);

freeb(&b2);

n++;

free(p3);

k = dtlong(vt2, index11, bvt2, index12, p4, index10);

free(ev2);

free(bev2);

}

printf("充電站位置：\n");

for (int i = 0; i < index10; i++) {

printf("(%.2f,%.2f)\n", p4[i].x, p4[i].y);

}

printf("總共需新增%d個充電站\n離充電站最遠的距離為%f公里\n點為：\n", n, k);

for (int i = 0; i < index11; i++) {

printf("(%.2f,%.2f)\n", vt2[i].p.x, vt2[i].p.y);

}

printf("邊界點為：\n");

for (int i = 0; i < index12; i++) {

printf("(%.2f,%.2f)\n", bvt2[i].p.x, bvt2[i].p.y);

}

print(bvt2, index12, vt2, index11);

END = clock();

printf("Time:%.0fms", (END - START));

delete[]ev;

delete[]bvt;

delete[]bev;

delete[]vt;

delete[]p;

delete[]p2;

free(vt2);

free(bvt2);

free(p4);

fclose(map);

system("pause");

return 0;

}

void freeb(bline\* b) {

if (\*b == NULL) {

return;

}

freeb(&((\*b)->rp));

free(\*b);

}

point dt(v\* vt, int\* index4, bv\* bvt, int\* index6, point\* p, int index, point\* p3, int\* index8) {

double max = 0;

point o;

int w = 0;

int oindex;

for (int i = 0; i < \*index4; i++) {

if (max <= pow(vt[i].p1.x - vt[i].p.x, 2) + pow(vt[i].p1.y - vt[i].p.y, 2)) {

bool ife2 = true;

for (int j = 0; j < index; j++) {

if (vt[i].p.x == p[j].x && vt[i].p.y == p[j].y) {

ife2 = false;

break;

}

}

if (ife2) {

max = pow(vt[i].p1.x - vt[i].p.x, 2) + pow(vt[i].p1.y - vt[i].p.y, 2);

o = vt[i].p;

oindex = i;

}

}

}

for (int i = 0; i < \*index6; i++) {

if (max <= pow(bvt[i].p1.x - bvt[i].p.x, 2) + pow(bvt[i].p1.y - bvt[i].p.y, 2)) {

bool ife2 = true;

for (int j = 0; j < index; j++) {

if (bvt[i].p.x == p[j].x && bvt[i].p.y == p[j].y) {

ife2 = false;

break;

}

}

if (ife2) {

max = pow(bvt[i].p1.x - bvt[i].p.x, 2) + pow(bvt[i].p1.y - bvt[i].p.y, 2);

o = bvt[i].p;

oindex = i;

w = 1;

}

}

}

double min = maxx \* maxx + maxy \* maxy;

bool ife = true;

for (int i = 0; i < index; i++) {

if (p[i].x == 0 && p[i].y == 0) {

ife = false;

break;

}

}

if (ife) {

for (int i = 0; i < index; i++) {

if (min >= pow(p[i].x, 2) + pow(p[i].y, 2)) {

min = pow(p[i].x, 2) + pow(p[i].y, 2);

}

}

if (max < min) {

max = min;

o.x = 0;

o.y = 0;

w = 2;

}

}

ife = true;

min = maxx \* maxx + maxy \* maxy;

for (int i = 0; i < index; i++) {

if (p[i].x == 0 && p[i].y == maxy) {

ife = false;

break;

}

}

if (ife) {

for (int i = 0; i < index; i++) {

if (min >= pow(p[i].x, 2) + pow(p[i].y - maxy, 2)) {

min = pow(p[i].x, 2) + pow(p[i].y - maxy, 2);

}

}

if (max < min) {

max = min;

o.x = 0;

o.y = maxy;

w = 3;

}

}

ife = true;

min = maxx \* maxx + maxy \* maxy;

for (int i = 0; i < index; i++) {

if (p[i].x == maxx && p[i].y == maxy) {

ife = false;

break;

}

}

if (ife) {

for (int i = 0; i < index; i++) {

if (min >= pow(p[i].x - maxx, 2) + pow(p[i].y - maxy, 2)) {

min = pow(p[i].x - maxx, 2) + pow(p[i].y - maxy, 2);

}

}

if (max <= min) {

max = min;

o.x = maxx;

o.y = maxy;

w = 4;

}

}

ife = true;

min = maxx \* maxx + maxy \* maxy;

for (int i = 0; i < index; i++) {

if (p[i].x == maxx && p[i].y == 0) {

ife = false;

break;

}

}

if (ife) {

for (int i = 0; i < index; i++) {

if (min >= pow(p[i].x - maxx, 2) + pow(p[i].y, 2)) {

min = pow(p[i].x - maxx, 2) + pow(p[i].y, 2);

}

}

if (max < min) {

max = min;

o.x = maxx;

o.y = 0;

w = 5;

}

}

insertv(o, p3, index8);

switch (w) {

case 0:

insertv(vt[oindex].p1, p3, index8);

insertv(vt[oindex].p2, p3, index8);

insertv(vt[oindex].p3, p3, index8);

dv(vt, oindex, index4);

break;

case 1:

insertv(bvt[oindex].p1, p3, index8);

insertv(bvt[oindex].p2, p3, index8);

dbv(bvt, oindex, index6);

break;

case 2:

for (int i = 0; i < index; i++) {

if (max == pow(p[i].x, 2) + pow(p[i].y, 2)) {

insertv(p[i], p3, index8);

}

}

break;

case 3:

for (int i = 0; i < index; i++) {

if (max == pow(p[i].x, 2) + pow(p[i].y - maxy, 2)) {

insertv(p[i], p3, index8);

}

}

break;

case 4:

for (int i = 0; i < index; i++) {

if (max == pow(p[i].x - maxx, 2) + pow(p[i].y - maxy, 2)) {

insertv(p[i], p3, index8);

}

}

break;

case 5:

for (int i = 0; i < index; i++) {

if (max == pow(p[i].x - maxx, 2) + pow(p[i].y, 2)) {

insertv(p[i], p3, index8);

}

}

break;

}

return o;

}

void checkdbv(point\* p, bv\* bvt, int index9, int\* index6, int index7) {

double min;

for (int i = index9; i < \*index6; i++) {

for (int j = 0; j < index9; j++) {

if (bvt[i].p.x == bvt[j].p.x && bvt[i].p.y == bvt[j].p.y) {

dbv(bvt, i--, index6);

break;

}

}

}

for (int i = index9; i < \*index6; i++) {

min = pow(bvt[i].p1.x - bvt[i].p.x, 2) + pow(bvt[i].p1.y - bvt[i].p.y, 2);

for (int j = 0; j < index7; j++) {

if (bvt[i].p2.x != p[j].x || bvt[i].p2.y != p[j].y) {

if (min > pow(p[j].x - bvt[i].p.x, 2) + pow(p[j].y - bvt[i].p.y, 2)) {

dbv(bvt, i--, index6);

break;

}

}

}

}

}

void checkdv(point\* p, v\* vt, int index15, int\* index4, int index7) {

double min;

for (int i = index15; i < \*index4; i++) {

for (int j = 0; j < index15; j++) {

if (vt[i].p.x == vt[j].p.x && vt[i].p.y == vt[j].p.y) {

dv(vt, i--, index4);

break;

}

}

}

for (int i = index15; i < \*index4; i++) {

min = pow(vt[i].p1.x - vt[i].p.x, 2) + pow(vt[i].p1.y - vt[i].p.y, 2);

for (int j = 0; j < index7; j++) {

if ((vt[i].p2.x != p[j].x || vt[i].p2.y != p[j].y) && (vt[i].p3.x != p[j].x || vt[i].p3.y != p[j].y)) {

if (min > pow(p[j].x - vt[i].p.x, 2) + pow(p[j].y - vt[i].p.y, 2)) {

dv(vt, i--, index4);

break;

}

}

}

}

}

double dtlong(v\* vt, int index4, bv\* bvt, int index6, point\* p, int index) {

double max = 0;

for (int i = 0; i < index4; i++) {

if (max <= pow(vt[i].p1.x - vt[i].p.x, 2) + pow(vt[i].p1.y - vt[i].p.y, 2)) {

bool ife2 = true;

for (int j = 0; j < index; j++) {

if (vt[i].p.x == p[j].x && vt[i].p.y == p[j].y) {

ife2 = false;

break;

}

}

if (ife2) {

max = pow(vt[i].p1.x - vt[i].p.x, 2) + pow(vt[i].p1.y - vt[i].p.y, 2);

}

}

}

for (int i = 0; i < index6; i++) {

if (max <= pow(bvt[i].p1.x - bvt[i].p.x, 2) + pow(bvt[i].p1.y - bvt[i].p.y, 2)) {

bool ife2 = true;

for (int j = 0; j < index; j++) {

if (bvt[i].p.x == p[j].x && bvt[i].p.y == p[j].y) {

ife2 = false;

break;

}

}

if (ife2) {

max = pow(bvt[i].p1.x - bvt[i].p.x, 2) + pow(bvt[i].p1.y - bvt[i].p.y, 2);

}

}

}

double min = maxx \* maxx + maxy \* maxy;

bool ife = true;

for (int i = 0; i < index; i++) {

if (p[i].x == 0 && p[i].y == 0) {

ife = false;

break;

}

}

if (ife) {

for (int i = 0; i < index; i++) {

if (min >= pow(p[i].x, 2) + pow(p[i].y, 2)) {

min = pow(p[i].x, 2) + pow(p[i].y, 2);

}

}

if (max < min) {

max = min;

}

}

ife = true;

min = maxx \* maxx + maxy \* maxy;

for (int i = 0; i < index; i++) {

if (p[i].x == 0 && p[i].y == maxy) {

ife = false;

break;

}

}

if (ife) {

for (int i = 0; i < index; i++) {

if (min >= pow(p[i].x, 2) + pow(p[i].y - maxy, 2)) {

min = pow(p[i].x, 2) + pow(p[i].y - maxy, 2);

}

}

if (max < min) {

max = min;

}

}

ife = true;

min = maxx \* maxx + maxy \* maxy;

for (int i = 0; i < index; i++) {

if (p[i].x == maxx && p[i].y == maxy) {

ife = false;

break;

}

}

if (ife) {

for (int i = 0; i < index; i++) {

if (min >= pow(p[i].x - maxx, 2) + pow(p[i].y - maxy, 2)) {

min = pow(p[i].x - maxx, 2) + pow(p[i].y - maxy, 2);

}

}

if (max <= min) {

max = min;

}

}

ife = true;

min = maxx \* maxx + maxy \* maxy;

for (int i = 0; i < index; i++) {

if (p[i].x == maxx && p[i].y == 0) {

ife = false;

break;

}

}

if (ife) {

for (int i = 0; i < index; i++) {

if (min >= pow(p[i].x - maxx, 2) + pow(p[i].y, 2)) {

min = pow(p[i].x - maxx, 2) + pow(p[i].y, 2);

}

}

if (max < min) {

max = min;

}

}

return sqrt(max);

}

void dv(v\* vt, int i, int\* index4) {

for (int j = i; j < \*index4 - 1; j++) {

vt[j] = vt[j + 1];

}

(\*index4)--;

}

void dbv(bv\* bvt, int i, int\* index6) {

for (int j = i; j < \*index6 - 1; j++) {

bvt[j] = bvt[j + 1];

}

(\*index6)--;

}

void insertv(point p, point\* p2, int\* index8) {

if (\*index8 == 0) {

p2[(\*index8)++] = p;

return;

}

bool ife = true;

for (int i = 0; i < \*index8; i++) {

if (p2[i].x == p.x && p2[i].y == p.y) {

ife = false;

break;

}

}

if (ife) {

if (p2[\*index8 - 1].y < p.y || (p2[\*index8 - 1].y == p.y && p2[\*index8 - 1].x <= p.x)) {

p2[(\*index8)++] = p;

}

else if (p2[0].y > p.y || (p2[0].x >= p.x && p2[0].y == p.y)) {

for (int i = \*index8; i > 0; i--) {

p2[i] = p2[i - 1];

}

p2[0] = p;

(\*index8)++;

}

else {

int x;

for (int i = 0; i < \*index8 - 1; i++) {

if (p2[i].y < p.y && p2[i + 1].y > p.y) {

x = i + 1;

break;

}

else if (p2[i].y == p.y && p2[i].x <= p.x && p2[i + 1].y == p.y && p2[i + 1].x >= p.x) {

x = i + 1;

break;

}

else if (p2[i].y == p.y && p2[i].x <= p.x && p2[i + 1].y > p.y) {

x = i + 1;

break;

}

else if (p2[i].y < p.y&& p2[i + 1].x >= p.x && p2[i + 1].y == p.y) {

x = i + 1;

break;

}

}

for (int i = \*index8; i > x; i--) {

p2[i] = p2[i - 1];

}

p2[x] = p;

(\*index8)++;

}

}

}

void print(bv\* bvt, int index6, v\* vt, int index4) {

bool ife = true;

for (int i = 0; i < index6; i++) {

if (bvt[i].p.x == 0 && bvt[i].p.y == 0) {

ife = false;

}

}

for (int i = 0; i < index4; i++) {

if (vt[i].p.x == 0 && vt[i].p.y == 0) {

ife = false;

}

}

if (ife) {

printf("(0.00,0.00)\n");

}

ife = true;

for (int i = 0; i < index6; i++) {

if (bvt[i].p.x == 0 && bvt[i].p.y == maxy) {

ife = false;

}

}

for (int i = 0; i < index4; i++) {

if (vt[i].p.x == 0 && vt[i].p.y == maxy) {

ife = false;

}

}

if (ife) {

printf("(0.00,%d.00)\n", maxy);

}

ife = true;

for (int i = 0; i < index6; i++) {

if (bvt[i].p.x == maxx && bvt[i].p.y == 0) {

ife = false;

}

}

for (int i = 0; i < index4; i++) {

if (vt[i].p.x == maxx && vt[i].p.y == 0) {

ife = false;

}

}

if (ife) {

printf("(%d.00,0.00)\n", maxx);

}

ife = true;

for (int i = 0; i < index6; i++) {

if (bvt[i].p.x == maxx && bvt[i].p.y == maxy) {

ife = false;

}

}

for (int i = 0; i < index4; i++) {

if (vt[i].p.x == maxx && vt[i].p.y == maxy) {

ife = false;

}

}

if (ife) {

printf("(%d.00,%d.00)\n", maxx, maxy);

}

}

point ppop(point\* p, int\* index) {

point p0 = p[0];

for (int i = 0; i < \*index - 1; i++) {

p[i] = p[i + 1];

}

(\*index)--;

return p0;

}

point ptop(point\* p) {

return p[0];

}

e\* epop(e\*\* ev, int\* index) {

e\* e0 = ev[0];

for (int i = 0; i < \*index - 1; i++) {

ev[i] = ev[i + 1];

}

(\*index)--;

return e0;

}

e\* etop(e\*\* ev) {

return ev[0];

}

be\* bepop(be\*\* bev, int\* index) {

be\* be0 = bev[0];

for (int i = 0; i < \*index - 1; i++) {

bev[i] = bev[i + 1];

}

(\*index)--;

return be0;

}

be\* betop(be\*\* bev) {

return bev[0];

}

void d(bline b) {

if (b->bev) {

b->bev->valid = false;

b->bev = NULL;

}

if (b->lp->bev) {

b->lp->bev->valid = false;

b->lp->bev = NULL;

}

if (b->bevd) {

b->bevd->valid = false;

b->bevd = NULL;

}

if (b->lp->bevd) {

b->lp->bevd->valid = false;

b->lp->bevd = NULL;

}

if (b->bevr) {

b->bevr->valid = false;

b->bevr = NULL;

}

if (b->lp->bevr) {

b->lp->bevr->valid = false;

b->lp->bevr = NULL;

}

if (b->bevl) {

b->bevl->valid = false;

b->bevl = NULL;

}

if (b->lp->bevl) {

b->lp->bevl->valid = false;

b->lp->bevl = NULL;

}

if (b->ev) {

b->ev->valid = false;

b->ev = NULL;

}

if (b->lp->ev) {

b->lp->ev->valid = false;

b->lp->ev = NULL;

}

if (b->rp->ev) {

b->rp->ev->valid = false;

b->rp->ev = NULL;

}

if (b->rp && b->lp) {

b->rp->lp = b->lp;

b->lp->rp = b->rp;

}

else if (b->rp) {

b->rp->lp = NULL;

}

else if (b->lp) {

b->lp->rp = NULL;

}

}

void process\_point(point\* p, int\* index, bline\* b, e\*\* ev, int\* index3, be\*\* bev, int\* index5) {

point p0 = ppop(p, index);

insert(p0, b, ev, index3, bev, index5);

}

void process\_event(e\*\* equeue, int\* index3, v\* v, int\* index4, be\*\* bequeue, int\* index5, bv\* bvt, int\* index6) {

e\* ev = epop(equeue, index3);

if (ev->valid) {

if (ev->o.x > maxx || ev->o.x < 0 || ev->o.y > maxy || ev->o.y < 0) {

}

else {

v[\*index4].p = ev->o;

v[\*index4].p2 = ev->b->p;

v[\*index4].p1 = ev->b->lp->p;

v[(\*index4)++].p3 = ev->b->rp->p;

}

d(ev->b);

boundu\_event(&ev->b->lp, bequeue, index5, ev->y);

boundd\_event(&ev->b->lp, bequeue, index5, ev->y);

boundr\_event(&ev->b->lp, bequeue, index5, ev->y);

boundl\_event(&ev->b->lp, bequeue, index5, ev->y);

circle\_event(&ev->b->lp, ev->y, equeue, index3);

circle\_event(&ev->b->rp, ev->y, equeue, index3);

if (ev->b->lp) {

free(ev->b);

}

else if (!ev->b->lp && ev->b->rp) {

ev->b = ev->b->rp;

free(ev->b->lp);

ev->b->lp = NULL;

}

else {

ev->b = NULL;

}

}

free(ev);

}

void process\_bevent(be\*\* bequeue, int\* index5, bv\* bv, int\* index6) {

be\* bev = bepop(bequeue, index5);

if (bev->valid) {

bv[\*index6].p = bev->o;

bv[\*index6].p2 = bev->b1->p;

bv[(\*index6)++].p1 = bev->b2->p;

if (bev->o.y == 0) {

bev->b1->bev = NULL;

}

else if (bev->o.y == maxy) {

bev->b1->bevd = NULL;

}

else if (bev->o.x == 0) {

bev->b1->bevl = NULL;

}

else if (bev->o.x == maxx) {

bev->b1->bevr = NULL;

}

}

free(bev);

}

void boundu\_event(bline\* b, be\*\* bequeue, int\* index5, double y0) {

if (!\*b || !(\*b)->rp) {

return;

}

double y;

point o;

be\* bev;

if (boundu((\*b)->p, (\*b)->rp->p, &y, &o) && y >= y0) {

if (o.x <= (\*b)->p.x) {

if (!(\*b)->lp || intersection((\*b)->lp->p, (\*b)->rp->p, y).y < o.y) {

}

else {

return;

}

}

else if (o.x >= (\*b)->rp->p.x) {

if (!(\*b)->rp->rp || intersection((\*b)->p, (\*b)->rp->rp->p, y).y < o.y) {

}

else {

return;

}

}

bev = (be\*)malloc(sizeof(\*bev));

bev->b1 = \*b;

bev->b2 = (\*b)->rp;

bev->o = o;

bev->valid = true;

bev->y = y;

(\*b)->bev = bev;

insertbe(bequeue, bev, index5);

}

}

void boundd\_event(bline\* b, be\*\* bequeue, int\* index5, double y0) {

if (!\*b || !(\*b)->rp) {

return;

}

double y;

point o;

be\* bev = NULL;

if (boundd((\*b)->p, (\*b)->rp->p, &y, &o) && y >= y0) {

bev = (be\*)malloc(sizeof(\*bev));

bev->b1 = \*b;

bev->b2 = (\*b)->rp;

bev->o = o;

bev->valid = true;

bev->y = y;

(\*b)->bevd = bev;

insertbe(bequeue, bev, index5);

}

}

void boundr\_event(bline\* b, be\*\* bequeue, int\* index5, double y0) {

if (!\*b || !(\*b)->rp) {

return;

}

double y;

point o;

be\* bev;

if (boundr((\*b)->p, (\*b)->rp->p, &y, &o) && y >= y0) {

bev = (be\*)malloc(sizeof(\*bev));

bev->b1 = \*b;

bev->b2 = (\*b)->rp;

bev->o = o;

bev->valid = true;

bev->y = y;

(\*b)->bevr = bev;

insertbe(bequeue, bev, index5);

}

}

void boundl\_event(bline\* b, be\*\* bequeue, int\* index5, double y0) {

if (!\*b || !(\*b)->rp) {

return;

}

double y;

point o;

be\* bev;

if (boundl((\*b)->p, (\*b)->rp->p, &y, &o) && y >= y0) {

bev = (be\*)malloc(sizeof(\*bev));

bev->b1 = \*b;

bev->b2 = (\*b)->rp;

bev->o = o;

bev->valid = true;

bev->y = y;

(\*b)->bevl = bev;

insertbe(bequeue, bev, index5);

}

}

bool boundl(point p1, point p2, double\* y, point\* o) {

if (p1.y >= p2.y) {

return false;

}

double oy = ((p1.x \* p1.x + p1.y \* p1.y) - (p2.x \* p2.x + p2.y \* p2.y)) / (2 \* (p1.y - p2.y));

if (oy < maxy && oy > 0) {

double y0 = (2 \* oy + sqrt(4 \* oy \* oy - 4 \* (2 \* oy \* p1.y - p1.x \* p1.x - p1.y \* p1.y))) / 2;

o->x = 0;

o->y = oy;

\*y = y0;

return true;

}

return false;

}

bool boundr(point p1, point p2, double\* y, point\* o) {

if (p1.y <= p2.y) {

return false;

}

double oy = (2 \* maxx \* (p2.x - p1.x) + p1.x \* p1.x + p1.y \* p1.y - p2.x \* p2.x - p2.y \* p2.y) / (2 \* (p1.y - p2.y));

if (oy < maxy && oy > 0) {

double y0 = (2 \* oy + sqrt(4 \* oy \* oy - 4 \* (2 \* oy \* p1.y + 2 \* p1.x \* maxx - maxx \* maxx - p1.x \* p1.x - p1.y \* p1.y))) / 2;

o->x = maxx;

o->y = oy;

\*y = y0;

return true;

}

return false;

}

bool boundd(point p1, point p2, double\* y, point\* o) {

if (p1.x == p2.x) {

return false;

}

double ox = (2 \* maxy \* (p2.y - p1.y) + p1.x \* p1.x + p1.y \* p1.y - p2.x \* p2.x - p2.y \* p2.y) / (2 \* (p1.x - p2.x));

if (ox <= maxy && ox >= 0) {

double y0 = (2 \* maxy + sqrt(4 \* maxy \* maxy - 4 \* (2 \* ox \* p1.x + 2 \* p1.y \* maxy - ox \* ox - p1.x \* p1.x - p1.y \* p1.y))) / 2;

o->x = ox;

o->y = maxy;

\*y = y0;

return true;

}

return false;

}

bool boundu(point p1, point p2, double\* y, point\* o) {

if (p1.x == p2.x) {

return false;

}

double ox = ((p1.x \* p1.x + p1.y \* p1.y) - (p2.x \* p2.x + p2.y \* p2.y)) / (2 \* (p1.x - p2.x));

if (ox <= maxx && ox >= 0) {

double y0 = sqrt(ox \* ox - 2 \* ox \* p1.x + p1.x \* p1.x + p1.y \* p1.y);

o->x = ox;

o->y = 0;

\*y = y0;

return true;

}

return false;

}

void inserte(e\*\* equeue, e\* ev, int\* index3) {

if (\*index3 == 0) {

equeue[0] = ev;

(\*index3)++;

return;

}

if (equeue[0]->y >= ev->y) {

for (int j = \*index3; j > 0; j--) {

equeue[j] = equeue[j - 1];

}

equeue[0] = ev;

(\*index3)++;

return;

}

if (equeue[\*index3 - 1]->y <= ev->y) {

equeue[\*index3] = ev;

(\*index3)++;

return;

}

for (int i = 0; i < \*index3 - 1; i++) {

if (equeue[i]->y <= ev->y && equeue[i + 1]->y >= ev->y) {

for (int j = \*index3; j > i + 1; j--) {

equeue[j] = equeue[j - 1];

}

equeue[i + 1] = ev;

(\*index3)++;

break;

}

}

}

void insertbe(be\*\* bequeue, be\* bev, int\* index5) {

if (\*index5 == 0) {

bequeue[0] = bev;

(\*index5)++;

return;

}

if (bequeue[0]->y >= bev->y) {

for (int j = \*index5; j > 0; j--) {

bequeue[j] = bequeue[j - 1];

}

bequeue[0] = bev;

(\*index5)++;

return;

}

if (bequeue[\*index5 - 1]->y <= bev->y) {

bequeue[\*index5] = bev;

(\*index5)++;

return;

}

for (int i = 0; i < \*index5 - 1; i++) {

if (bequeue[i]->y <= bev->y && bequeue[i + 1]->y >= bev->y) {

for (int j = \*index5; j > i + 1; j--) {

bequeue[j] = bequeue[j - 1];

}

bequeue[i + 1] = bev;

(\*index5)++;

break;

}

}

}

void circle\_event(bline\* b, double y0, e\*\* equeue, int\* index3) {

if (!\*b || !(\*b)->lp || !(\*b)->rp) {

return;

}

double y;

point o;

e\* ev;

if (circle((\*b)->lp->p, (\*b)->p, (\*b)->rp->p, &y, &o) && y >= y0) {

ev = (e\*)malloc(sizeof(\*ev));

ev->valid = true;

ev->y = y;

ev->o = o;

ev->b = \*b;

(\*b)->ev = ev;

inserte(equeue, ev, index3);

}

}

bool circle(point a, point b, point c, double\* y, point\* o) {

if ((b.y - a.y) \* (c.x - a.x) - (c.y - a.y) \* (b.x - a.x) > 0) {

return false;

}

double A = 2 \* (b.x - a.x), B = 2 \* (b.y - a.y),

A2 = 2 \* (c.x - b.x), B2 = 2 \* (c.y - b.y),

C = b.x \* b.x + b.y \* b.y - a.x \* a.x - a.y \* a.y,

C2 = c.x \* c.x + c.y \* c.y - b.x \* b.x - b.y \* b.y,

D = (A \* B2 - A2 \* B);

if (D == 0) {

return false;

}

o->x = (B2 \* C - B \* C2) / D;

o->y = (A \* C2 - A2 \* C) / D;

\*y = o->y + sqrt(pow(a.y - o->y, 2) + pow(a.x - o->x, 2));

return true;

}

void insertion(point p, bline\* b) {

if ((\*b)->bev) {

(\*b)->bev->valid = false;

(\*b)->bev = NULL;

}

if ((\*b)->bevd) {

(\*b)->bevd->valid = false;

(\*b)->bevd = NULL;

}

if ((\*b)->bevr) {

(\*b)->bevr->valid = false;

(\*b)->bevr = NULL;

}

if ((\*b)->bevl) {

(\*b)->bevl->valid = false;

(\*b)->bevl = NULL;

}

if ((\*b)->ev) {

(\*b)->ev->valid = false;

(\*b)->ev = NULL;

}

if ((\*b)->rp && (\*b)->rp->ev) {

(\*b)->rp->ev->valid = false;

(\*b)->rp->ev = NULL;

}

bline i, j = \*b;

i = (bline)malloc(sizeof(\*i));

i->p = p;

i->lp = i->rp = NULL;

i->ev = NULL;

i->bev = NULL;

i->bevd = NULL;

i->bevr = NULL;

i->bevl = NULL;

i->lp = j;

if ((\*b)->rp) {

i->rp = j->rp;

j->rp->lp = i;

}

j->rp = i;

}

void insert(point p, bline\* b, e\*\* ev, int\* index3, be\*\* bev, int\* index5) {

if (!\*b) {

bline a = NULL;

a = (bline)malloc(sizeof(\*a));

a->p = p;

a->lp = a->rp = NULL;

a->ev = NULL;

a->bev = NULL;

a->bevd = NULL;

a->bevr = NULL;

a->bevl = NULL;

\*b = a;

return;

}

bool iftwo = false;

int j = 0;

for (bline i = \*b; i; i = i->rp, j++) {

if (ifintersect(p, i)) {

if (j == 0) {

if (intersection(p, i->p, p.y).y > 0) {

bline a;

a = (bline)malloc(sizeof(\*a));

a->p = i->p;

a->lp = a->rp = NULL;

a->ev = NULL;

a->bev = NULL;

a->bevd = NULL;

a->bevr = NULL;

a->bevl = NULL;

a->rp = i;

i->lp = a;

insertion(p, &i->lp);

boundu\_event(&i->lp, bev, index5, p.y);

boundu\_event(&i->lp->lp, bev, index5, p.y);

boundr\_event(&i->lp, bev, index5, p.y);

boundr\_event(&i->lp->lp, bev, index5, p.y);

boundl\_event(&i->lp, bev, index5, p.y);

boundl\_event(&i->lp->lp, bev, index5, p.y);

boundd\_event(&i->lp, bev, index5, p.y);

boundd\_event(&i->lp->lp, bev, index5, p.y);

circle\_event(&i, p.y, ev, index3);

circle\_event(&i->lp, p.y, ev, index3);

\*b = i->lp->lp;

}

else if (intersection(p, i->p, p.y).x > i->p.x) {

insertion(p, &i);

boundu\_event(&i, bev, index5, p.y);

boundu\_event(&i->rp, bev, index5, p.y);

boundd\_event(&i, bev, index5, p.y);

boundd\_event(&i->rp, bev, index5, p.y);

boundr\_event(&i, bev, index5, p.y);

boundr\_event(&i->rp, bev, index5, p.y);

boundl\_event(&i, bev, index5, p.y);

boundl\_event(&i->rp, bev, index5, p.y);

circle\_event(&i->rp, p.y, ev, index3);

circle\_event(&i->rp->rp, p.y, ev, index3);

}

else {

bline a;

a = (bline)malloc(sizeof(\*a));

a->p = p;

a->lp = a->rp = NULL;

a->ev = NULL;

a->bev = NULL;

a->bevd = NULL;

a->bevr = NULL;

a->bevl = NULL;

a->rp = i;

i->lp = a;

boundu\_event(&i->lp, bev, index5, p.y);

boundd\_event(&i->lp, bev, index5, p.y);

boundr\_event(&i->lp, bev, index5, p.y);

boundl\_event(&i->lp, bev, index5, p.y);

circle\_event(&i, p.y, ev, index3);

\*b = i->lp;

}

return;

}

if (!i->rp) {

if (intersection(i->p, p, p.y).y > 0) {

insertion(i->p, &i);

insertion(p, &i);

boundu\_event(&i, bev, index5, p.y);

boundu\_event(&i->rp, bev, index5, p.y);

boundd\_event(&i, bev, index5, p.y);

boundd\_event(&i->rp, bev, index5, p.y);

boundr\_event(&i, bev, index5, p.y);

boundr\_event(&i->rp, bev, index5, p.y);

boundl\_event(&i, bev, index5, p.y);

boundl\_event(&i->rp, bev, index5, p.y);

circle\_event(&i, p.y, ev, index3);

circle\_event(&i->rp, p.y, ev, index3);

}

else if (intersection(p, i->p, p.y).x > i->p.x) {

insertion(p, &i);

boundu\_event(&i, bev, index5, p.y);

boundd\_event(&i, bev, index5, p.y);

boundr\_event(&i, bev, index5, p.y);

boundl\_event(&i, bev, index5, p.y);

circle\_event(&i, p.y, ev, index3);

}

else {

insertion(p, &i->lp);

boundu\_event(&i->lp, bev, index5, p.y);

boundu\_event(&i->lp->lp, bev, index5, p.y);

boundr\_event(&i->lp, bev, index5, p.y);

boundr\_event(&i->lp->lp, bev, index5, p.y);

boundd\_event(&i->lp, bev, index5, p.y);

boundd\_event(&i->lp->lp, bev, index5, p.y);

boundl\_event(&i->lp, bev, index5, p.y);

boundl\_event(&i->lp->lp, bev, index5, p.y);

circle\_event(&i->lp, p.y, ev, index3);

circle\_event(&i->lp->lp, p.y, ev, index3);

}

return;

}

if (i->rp && intersection(p, i->rp->p, p.y).y < intersection(p, i->p, p.y).y) {

insertion(i->p, &i);

iftwo = true;

}

insertion(p, &i);

boundu\_event(&i, bev, index5, p.y);

boundu\_event(&i->rp, bev, index5, p.y);

boundd\_event(&i, bev, index5, p.y);

boundd\_event(&i->rp, bev, index5, p.y);

boundr\_event(&i, bev, index5, p.y);

boundr\_event(&i->rp, bev, index5, p.y);

boundl\_event(&i, bev, index5, p.y);

boundl\_event(&i->rp, bev, index5, p.y);

circle\_event(&i->rp, p.y, ev, index3);

circle\_event(&i, p.y, ev, index3);

circle\_event(&i->rp->rp, p.y, ev, index3);

if (iftwo) {

boundu\_event(&i->rp->rp, bev, index5, p.y);

boundd\_event(&i->rp->rp, bev, index5, p.y);

boundl\_event(&i->rp->rp, bev, index5, p.y);

boundr\_event(&i->rp->rp, bev, index5, p.y);

circle\_event(&i->rp->rp->rp, p.y, ev, index3);

}

return;

}

}

insertion(p, b);

boundu\_event(b, bev, index5, p.y);

boundr\_event(b, bev, index5, p.y);

boundl\_event(b, bev, index5, p.y);

boundd\_event(b, bev, index5, p.y);

}

bool ifintersect(point p1, bline p2) {

if (p2->p.y == p1.y) {

return false;

}

double a, b;

if (p2->lp) {

a = intersection(p2->lp->p, p2->p, p1.y).x;

}

if (p2->rp) {

b = intersection(p2->p, p2->rp->p, p1.y).x;

}

if ((!p2->lp || a <= p1.x) && (!p2->rp || p1.x <= b)) {

return true;

}

return false;

}

point intersection(point p1, point p2, double d) {//找斷點

point p;

if (p1.y == p2.y) {

p.x = (p1.x + p2.x) / 2;

}

else if (p2.y == d) {

p.x = p2.x;

}

else if (p1.y == d) {

p.x = p1.x;

p1 = p2;

}

else {

double z0 = 2 \* (p1.y - d);

double z1 = 2 \* (p2.y - d);

double a = 1 / z0 - 1 / z1;

double b = -2 \* (p1.x / z0 - p2.x / z1);

double c = (p1.x \* p1.x + p1.y \* p1.y - d \* d) / z0 - (p2.x \* p2.x + p2.y \* p2.y - d \* d) / z1;

double x1 = (-b - sqrt(b \* b - 4 \* a \* c)) / (2 \* a);

double x2 = (-b + sqrt(b \* b - 4 \* a \* c)) / (2 \* a);

if (p1.x == p2.x) {

if (p1.y < p2.y) {

if (x1 < x2) {

p.x = x1;

}

else {

p.x = x2;

}

}

else {

if (x1 < x2) {

p.x = x2;

}

else {

p.x = x1;

}

}

}

else if (p1.x > p2.x) {

if (p1.y < p2.y) {

if (x1 < x2) {

p.x = x1;

}

else {

p.x = x2;

}

}

else {

if (x1 < x2) {

p.x = x2;

}

else {

p.x = x1;

}

}

}

else {

if (p1.y < p2.y) {

if (x1 < x2) {

p.x = x1;

}

else {

p.x = x2;

}

}

else {

if (x1 < x2) {

p.x = x2;

}

else {

p.x = x1;

}

}

}

}

p.y = (p1.y \* p1.y + (p1.x - p.x) \* (p1.x - p.x) - d \* d) / (2 \* p1.y - 2 \* d);

return p;

}

2. Input Code Format

Three of examples for input use are in below….

\*因字數過大佔篇幅，所以放在檔案中\*

(1)map1.txt

(2)map2.txt

(3)map3.txt

3. Output Code Format

Three of examples for output use are in below….

\*因字數過大佔篇幅，所以放在檔案中\*

(1)o1.txt

(2)o2.txt

(3)o3.txt

**(五) 執行結果、討論與心得**

執行結果與討論 (執行時間、problem *n*的大小等問題討論)等…

**(1) 依據充電站的位置(2D)，建構一個Voronoi Diagram with time complexity O(𝑛log𝑛).**

1. 執行結果

Output of program:

\*因字數過大佔篇幅，所以放在檔案中\*

1. o1.txt
2. o2.txt
3. o3.txt
4. o4.txt
5. o5.txt
6. o6.txt
7. o7.txt
8. o8.txt
9. o9.txt
10. o10.txt
11. o11.txt
12. o12.txt
13. o13.txt
14. o14.txt
15. o15.txt
16. o16.txt
17. o17.txt
18. o18.txt
19. o19.txt
20. o20.txt

2. 討論

執行時間、問題大小等問題討論

1. Running Time
2. 663ms
3. 697ms
4. 778ms
5. 875ms
6. 961ms
7. 1204ms
8. 1151ms
9. 1791ms
10. 1867ms
11. 2419ms
12. 316ms
13. 607ms
14. 1410ms
15. 2554ms
16. 3814ms
17. 5477ms
18. 7375ms
19. 9631ms
20. 13170ms
21. 15612ms

(2) Problem size *n*

1. 有50個充電站的1000\*1000的地圖
2. 有100個充電站的1000\*1000的地圖
3. 有150個充電站的1000\*1000的地圖
4. 有200個充電站的1000\*1000的地圖
5. 有250個充電站的1000\*1000的地圖
6. 有300個充電站的1000\*1000的地圖
7. 有350個充電站的1000\*1000的地圖
8. 有400個充電站的1000\*1000的地圖
9. 有450個充電站的1000\*1000的地圖
10. 有500個充電站的1000\*1000的地圖
11. 有10個充電站的500\*500的地圖
12. 有10個充電站的1000\*1000的地圖
13. 有10個充電站的1500\*1500的地圖
14. 有10個充電站的2000\*2000的地圖
15. 有10個充電站的2500\*2500的地圖
16. 有10個充電站的3000\*3000的地圖
17. 有10個充電站的3500\*3500的地圖
18. 有10個充電站的4000\*4000的地圖
19. 有10個充電站的4500\*4500的地圖
20. 有10個充電站的5000\*5000的地圖

O(a\*b)+O(nlogn)，第一張符合O(nlogn)，第二張符合O(a\*b)

**(2) 依序列出充電站的服務範圍(面積)。**

1. 執行結果

Output of program:

\*因字數過大佔篇幅，所以放在檔案中\*

1. o1.txt
2. o2.txt
3. o3.txt
4. o4.txt
5. o5.txt
6. o6.txt
7. o7.txt
8. o8.txt
9. o9.txt
10. o10.txt
11. o11.txt
12. o12.txt
13. o13.txt
14. o14.txt
15. o15.txt
16. o16.txt
17. o17.txt
18. o18.txt
19. o19.txt
20. o20.txt

2. 討論

執行時間、問題大小等問題討論

1. Running Time
2. 849ms
3. 807ms
4. 878ms
5. 958ms
6. 1210ms
7. 1397ms
8. 1648ms
9. 2218ms
10. 3375ms
11. 3733ms
12. 347ms
13. 682ms
14. 1423ms
15. 2509ms
16. 3980ms
17. 5556ms
18. 7550ms
19. 10072ms
20. 12466ms
21. 15938ms

(2) Problem size *n*

1. 有50個充電站的1000\*1000的地圖
2. 有100個充電站的1000\*1000的地圖
3. 有150個充電站的1000\*1000的地圖
4. 有200個充電站的1000\*1000的地圖
5. 有250個充電站的1000\*1000的地圖
6. 有300個充電站的1000\*1000的地圖
7. 有350個充電站的1000\*1000的地圖
8. 有400個充電站的1000\*1000的地圖
9. 有450個充電站的1000\*1000的地圖
10. 有500個充電站的1000\*1000的地圖
11. 有10個充電站的500\*500的地圖
12. 有10個充電站的1000\*1000的地圖
13. 有10個充電站的1500\*1500的地圖
14. 有10個充電站的2000\*2000的地圖
15. 有10個充電站的2500\*2500的地圖
16. 有10個充電站的3000\*3000的地圖
17. 有10個充電站的3500\*3500的地圖
18. 有10個充電站的4000\*4000的地圖
19. 有10個充電站的4500\*4500的地圖
20. 有10個充電站的5000\*5000的地圖

O(a\*b)+O(n3)，第一張符合O(n3)，第二張符合O(a\*b)

**(3) 找出離充電站最遠的點以及距離(Largest Empty Circle Problem with Voronoi Diagram, also called toxic waste dump problem)。**

1. 執行結果

Output of program:

\*因字數過大佔篇幅，所以放在檔案中\*

1. o1.txt
2. o2.txt
3. o3.txt
4. o4.txt
5. o5.txt
6. o6.txt
7. o7.txt
8. o8.txt
9. o9.txt
10. o10.txt
11. o11.txt
12. o12.txt
13. o13.txt
14. o14.txt
15. o15.txt
16. o16.txt
17. o17.txt
18. o18.txt
19. o19.txt
20. o20.txt

2. 討論

執行時間、問題大小等問題討論

1. Running Time
2. 685ms
3. 742ms
4. 791ms
5. 947ms
6. 1181ms
7. 1488ms
8. 1919ms
9. 2833ms
10. 3254ms
11. 3275ms
12. 310ms
13. 655ms
14. 1391ms
15. 2507ms
16. 3817ms
17. 5475ms
18. 7498ms
19. 9746ms
20. 12647ms
21. 15397ms

(2) Problem size *n*

1. 有50個充電站的1000\*1000的地圖
2. 有100個充電站的1000\*1000的地圖
3. 有150個充電站的1000\*1000的地圖
4. 有200個充電站的1000\*1000的地圖
5. 有250個充電站的1000\*1000的地圖
6. 有300個充電站的1000\*1000的地圖
7. 有350個充電站的1000\*1000的地圖
8. 有400個充電站的1000\*1000的地圖
9. 有450個充電站的1000\*1000的地圖
10. 有500個充電站的1000\*1000的地圖
11. 有10個充電站的500\*500的地圖
12. 有10個充電站的1000\*1000的地圖
13. 有10個充電站的1500\*1500的地圖
14. 有10個充電站的2000\*2000的地圖
15. 有10個充電站的2500\*2500的地圖
16. 有10個充電站的3000\*3000的地圖
17. 有10個充電站的3500\*3500的地圖
18. 有10個充電站的4000\*4000的地圖
19. 有10個充電站的4500\*4500的地圖
20. 有10個充電站的5000\*5000的地圖

O(a\*b)+O(nlogn)，第一張符合O(nlogn)，第二張符合O(a\*b)

**(4) 根據Sub-problem (3)，在該位置安裝一個充電站，則須在原來的Voronoi Diagram外，加一個點變成有*n+1*個點的Voronoi Diagram，設計一程式法建構該新的Voronoi Diagram。**

1. 執行結果

Output of program:

\*因字數過大佔篇幅，所以放在檔案中\*

1. o1.txt
2. o2.txt
3. o3.txt
4. o4.txt
5. o5.txt
6. o6.txt
7. o7.txt
8. o8.txt
9. o9.txt
10. o10.txt
11. o11.txt
12. o12.txt
13. o13.txt
14. o14.txt
15. o15.txt
16. o16.txt
17. o17.txt
18. o18.txt
19. o19.txt
20. o20.txt

2. 討論

執行時間、問題大小等問題討論

1. Running Time
2. 772ms
3. 820ms
4. 924ms
5. 1138ms
6. 1257ms
7. 1394ms
8. 1614ms
9. 2070ms
10. 3934ms
11. 4177ms
12. 354ms
13. 627ms
14. 1611ms
15. 2554ms
16. 3827ms
17. 5545ms
18. 7548ms
19. 10059ms
20. 12354ms
21. 15482ms

(2) Problem size *n*

1. 有50個充電站的1000\*1000的地圖
2. 有100個充電站的1000\*1000的地圖
3. 有150個充電站的1000\*1000的地圖
4. 有200個充電站的1000\*1000的地圖
5. 有250個充電站的1000\*1000的地圖
6. 有300個充電站的1000\*1000的地圖
7. 有350個充電站的1000\*1000的地圖
8. 有400個充電站的1000\*1000的地圖
9. 有450個充電站的1000\*1000的地圖
10. 有500個充電站的1000\*1000的地圖
11. 有10個充電站的500\*500的地圖
12. 有10個充電站的1000\*1000的地圖
13. 有10個充電站的1500\*1500的地圖
14. 有10個充電站的2000\*2000的地圖
15. 有10個充電站的2500\*2500的地圖
16. 有10個充電站的3000\*3000的地圖
17. 有10個充電站的3500\*3500的地圖
18. 有10個充電站的4000\*4000的地圖
19. 有10個充電站的4500\*4500的地圖
20. 有10個充電站的5000\*5000的地圖

O(a\*b)+O(nlogn)，第一張符合O(nlogn)，第二張符合O(a\*b)

**(5) 重複Sub-problems (3) and (4)直到最遠距離滿足一個門檻值(Threshold value *k*, 如500m，*k*值可自行設定)，並輸出總共需要幾個充電站、目前離充電站最遠的距離與*k*值。**

1. 執行結果

Output of program:

\*因字數過大佔篇幅，所以放在檔案中\*

1. o1.txt
2. o2.txt
3. o3.txt
4. o4.txt
5. o5.txt
6. o6.txt
7. o7.txt
8. o8.txt
9. o9.txt
10. o11.txt
11. o11.txt
12. o11.txt
13. o11.txt
14. o11.txt
15. o11.txt
16. o11.txt
17. o11.txt
18. o11.txt
19. o11.txt

2. 討論

執行時間、問題大小等問題討論

1. Running Time
2. 262ms
3. 403ms
4. 514ms
5. 581ms
6. 721ms
7. 1151ms
8. 2523ms
9. 4323ms
10. 8078ms
11. 17527ms
12. 197ms
13. 78ms
14. 70ms
15. 24ms
16. 27ms
17. 31ms
18. 26ms
19. 17ms
20. 18ms

(2) Problem size *n*

1. 有50個充電站的1000\*1000的地圖，k=100公里
2. 有100個充電站的1000\*1000的地圖，k=100公里
3. 有150個充電站的1000\*1000的地圖，k=100公里
4. 有200個充電站的1000\*1000的地圖，k=100公里
5. 有250個充電站的1000\*1000的地圖，k=100公里
6. 有300個充電站的1000\*1000的地圖，k=100公里
7. 有350個充電站的1000\*1000的地圖，k=100公里
8. 有400個充電站的1000\*1000的地圖，k=100公里
9. 有450個充電站的1000\*1000的地圖，k=100公里
10. 有10個充電站的1000\*1000的地圖，k=50公里
11. 有10個充電站的1000\*1000的地圖，k=100公里
12. 有10個充電站的1000\*1000的地圖，k=150公里
13. 有10個充電站的1000\*1000的地圖，k=200公里
14. 有10個充電站的1000\*1000的地圖，k=250公里
15. 有10個充電站的1000\*1000的地圖，k=300公里
16. 有10個充電站的1000\*1000的地圖，k=350公里
17. 有10個充電站的1000\*1000的地圖，k=400公里
18. 有10個充電站的1000\*1000的地圖，k=450公里
19. 有10個充電站的1000\*1000的地圖，k=500公里

因為我會分配m= n+round(a \* b / (k + 1) / (k + 1) + 1)的記憶體給最後總充電站數，以及各C(m,3)給vertex事件、邊界的vertex、vertex和邊界的vertex

所以當k太小或n太大時，可能會無法分配那麼多記憶體來計算，導致跑不出來。

所以當跑不出來，還請調大k值。

兩張皆符合O(n3logn/k2)

3. 心得

這次的作業題目其實不難理解，可是實踐非常的難，因為情況種類很多，要全面的考慮到其實並不容易，只能一步一步盡量接近完美。

一開始寫了一堆不能用的垃圾，後來決定看看別人怎麼寫的，結果又找到有錯的程式碼，但是好消息是，他的很多思考邏輯是值得學習的，雖然細節沒考慮到，但是大致方向仍然給了我很多靈感和幫助，讓我啟動的比較順利。

中途也遇到很多問題，大多是細節沒考慮到，導致不同狀況會出問題，也耗時了三周才完成，嘔心瀝血的作品也不算愧對本心。

但是要說完全沒錯，其實我自己也不敢保證，只能確定大多數狀況是對的，為了偵錯一直畫點，畫到都快爛了，做到後面開始心浮氣躁，還好最後還是戰勝了，至少做出了一定的成果，自己也算滿意。

這次也複習了linked list的用法，因為之前資料結構沒學好，所以花費了比較多心力在上面。

我從來都沒像現在這麼了解拋物線，導了很多公式，也了解很多拋物線的狀況。**參考文獻**

1. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest and Clifford Stein, "Introduction to Algorithms," Third Edition, The MIT Press, 2009.
2. R.C.T. Lee, S.S. Tseng, R.C. Chang, and Y.T.Tsai, "Introduction to the Design and Analysis of Algorithms," McGraw-Hill, 2005.
3. Anany V. Levitin, "Introduction to the Design and Analysis of Algorithms," 2nd Edition, Addison Wesley, 2007.
4. Richard Neapolitan and Kumarss Naimipour, "Foundations of Algorithms," Fourth Edition, Jones and Bartlett Publishers, 2010.
5. <https://www.cs.hmc.edu/~mbrubeck/voronoi.html>
6. <https://tedsieblog.wordpress.com/2018/11/29/voronoi-diagram-a-case-study-of-fortunes-algorithm/>