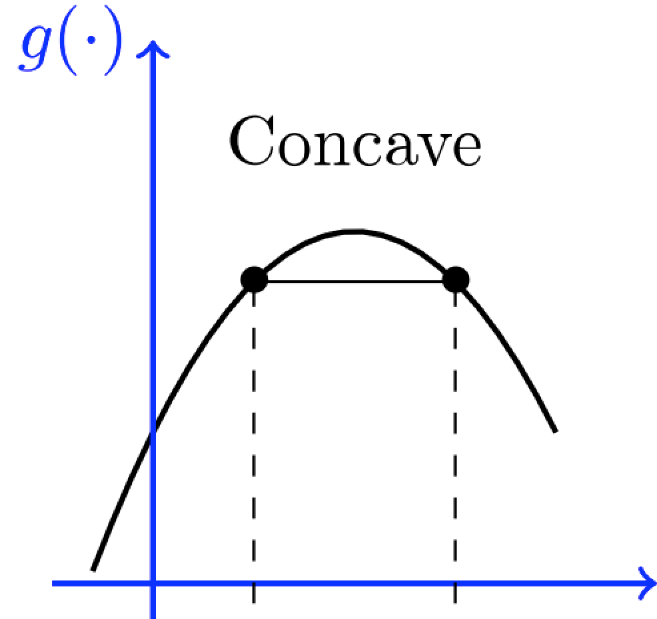
1. **Mở đầu**

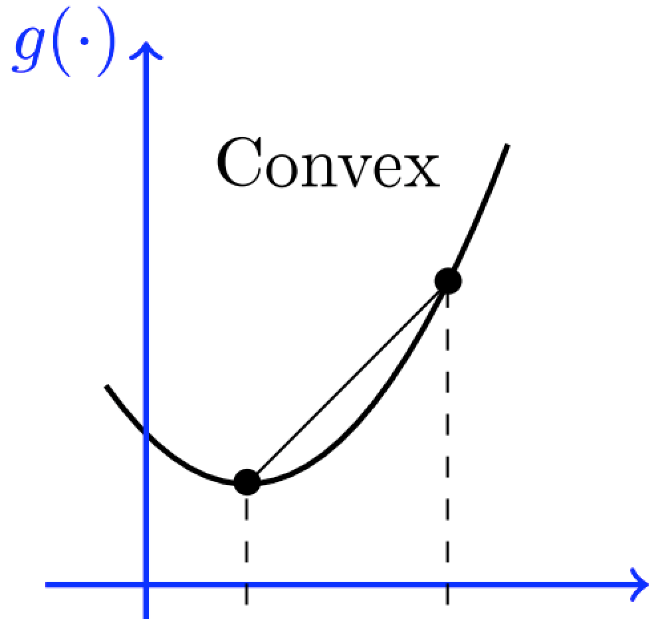
Cho một hàm F(x) chỉ có một cực trị duy nhất (unimodal). Có hai dạng hàm F(x) cơ bản:

* Phần đầu tăng chặt, đạt đến giá trị lớn nhất, sau đó giảm chặt. (concave)



Một hàm số thoả mãn tính chất này nếu tất cả các đoạn thẳng nối 2 điểm của đồ thị hàm số, nằm "bên dưới" của đồ thị.

* Phần đầu giảm chặt, đạt đến giá trị nhỏ nhất, sau đó tăng chặt. (convex)



Một hàm số thoả mãn tính chất này nếu tất cả các đoạn thẳng nối 2 điểm của đồ thị hàm số, đều nằm "bên trên" của đồ thị.

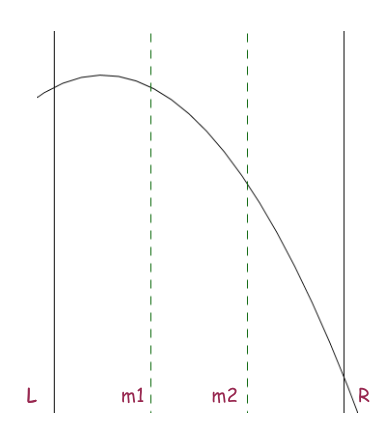
1. **Bài toán**

Cho một hàm F(x)F(x) trong đoạn [l,r][l,r] thoả mãn: F tăng chặt tới một cực đại (điểm H) rồi giảm chặt. Yêu cầu tìm điểm đạt giá trị lớn nhất (điểm H).

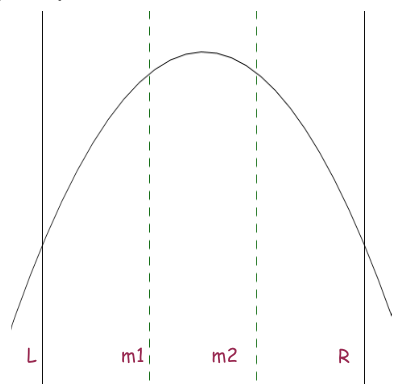
1. **Thuật toán**

Xét hai vị trí m1 và m2 trong đoạn [l,r] sao cho l<m1<m2<r. Rõ ràng cực trị có thể nằm ở 1 trong 3 phần:

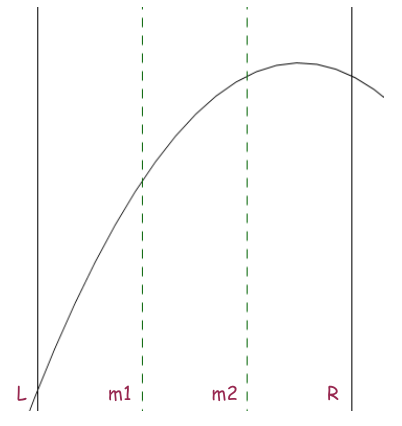
* [l,m1]. Khi đó, ta biết chắc chắn F(m1)>F(m2).



* [m1,m2]. Ta không thể rút ra kết luận gì về F(m1) và F(m2).



* [m2,R]. Tương tự trường hợp đầu, ta biết chắc chắn F(m1)<F(m2).



Ngược lại, bằng việc so sánh F(m1) và F(m2), ta có thể rút ra kết luận như sau:

* Nếu F(m1)<F(m2): Ta biết chắc chắn H nằm trong [m1,r][m1,r].
* F(m1)>F(m2): Ta biết chắc chắn H nằm trong [l,m2][l,m2].
* F(m1)=F(m2): H nằm trong [m1,m2]. (Chú ý: khi cài đặt chặt tam phân với hàm số thực, ta thường bỏ qua trường hợp này, để tránh sai số, và do trên thực tế 2 số thực hầu như không bao giờ bằng nhau).

Do đó, dựa vào việc so sánh F ở hai điểm m1, m2 ta có thể thay đổi và giảm không gian tìm kiếm [l,r] xuống một khoản không gian nhỏ hơn [l′,r′]. Nếu ta chọn:

* m1=l+(r−l)/3
* m2=r−(r−l)/3

Thì sau mỗi lần, độ lớn của đoạn [l,r] giảm xuống còn 2/3 lần.

Nếu ta lặp đi lặp lại K lần, thì độ lớn của [l, r] sẽ chỉ còn (2/3)K. Ví dụ với l=−109,r=109, ta lặp lại 100 lần, thì đoạn [l, r] thu về chỉ còn độ dài là (2/3.0)100∗(2∗109)<5∗109<5∗10-9, đủ chính xác với hầu hết mọi bài toán.

Độ phức tạp thuật toán là O(logT) với T là độ chính xác mà ta cần thực hiện.

# Cài đặt

**double** **max\_f**(**double** left, **double** right) {

**int** N\_ITER **=** 100;

**for** (**int** i **=** 0; i **<** N\_ITER; i**++**) {

**double** x1 **=** left **+** (right **-** left) **/** 3.0;

**double** x2 **=** right **-** (right **-** left) **/** 3.0;

**if** (f(x1) **>** f(x2)) right **=** x2;

**else** left **=** x1;

}

**return** f(left);

}

# Bài toán áp dụng

# Bài 1: Cuộc đua

Lập trình viên ith bắt đầu cuộc đua của mình tại một khoảng cách Di trước vạch xuất phát lúc T = 0. Tốc độ chiếc xe của lập trình viên ith bằng Si. Sau 1 thời gian là T, Vị trí của lập trình viên thứ ith sẽ là Pi(T) = Si\*T+Di. Hãy xác định hàm f(T)= max(Pi(T))-min(Pj(T)). Cuộc đua kết thúc lúc T=K. bạn cần tìm ra giá trị nhỏ nhất của hàm f(T) trong lúc đua.

- Dữ liệu vào:

Dòng đầu tiên chứa hai số nguyên N và K biểu thị số lượng bộ mã hóa và thời lượng của cuộc đua. (1 ≤ N, K ≤ 105)

N dòng tiếp theo, mỗi dòng chứa 2 số nguyên Si và Di (1 ≤ Si ≤ 105, 0 ≤ Di ≤ 105)

- Dữ liệu ra: In giá trị nhỏ nhất của hàm f(T) với độ chính xác sau dấu

phẩy 6 chữ số thập phân

## **Ý tưởng và thuật toán:**

Với mỗi thời gian T sẽ có 1 vị trí của các xe và ta sẽ tại thời điểm T đó thì cũng có chiều dài của dàn xe, với mỗi TTt thì chiều dài của dàn xe khác nhau, nhiệm vụ của chúng ta tìm ra chiều dài nhỏ nhất của dàn xe thông qua hàm xử lý f() với f() là hàm lõm. Như vậy ta đủ điệu kiện để sử dụng **Ternary search**

Ta sẽ tam phân trên KHOẢNG thời gian từ 0 đến K với hàm xử lý đạt được kết quả nhỏ nhất để thõa mãn yêu cầu bài toán. Ở hàm xử lý, với mỗi thời gian t sau khi khi đã chặt thành phân đoạn ở bước tam phân, sẽ tính quảng đường của mỗi xe(Pi(T) = Si\*T+Di), sau đó tìm quãng đường nhỏ nhất và lớn nhất trong các xe ở thời điểm t. Và cuối cùng tính trị tuyết đối của 2 quãng đường lớn nhất và nhỏ nhất vừa tìm được. Sau khi tam phân 100 lần thì sẽ cho ra kết quả của bài toán

Thuật toán:

B1: Gán L= 0 và R=K và tìm vị trí mid1 và mid2 qua công thức mid1=L+(R-L)/3, mid=R-(R-L)/3

B2: Tính F(mid1) và F(mid2) với hàm xử lý F(T):

Duyệt qua mỗi xe trong mảng từ i=0->n

(Pi(T) = Si\*T+Di)

Tìm max và min của Pi(T) ở mỗi xe, và cuối cùng trả về max-min

B3: So sánh F(mid1) và F(mid2) nếu:

F(mid1) > F(mid2) L= mid1 ngược lại R= mid2

B4: Lặp các bước trên 100 lần sẽ ra kết quả bài toán

# Bài 2: Giải cứu

Hôm nay là một ngày rất nóng đối với người cứu hộ Bob. Và ngay bây giờ anh mới nhận ra người đàn ông say rượu mới chìm dưới biển! Giả sử rằng khu nghỉ mát là mặt phẳng Euclide và trên bờ là đường y < 0. Biển là y>= 0. Tốc độ của Bob trên đất bằng với v1 và tốc độ của anh ta khi bơi bằng với v2. Bây giờ Bob đang ở điểm (x1, y1) và người đàn ông chìm ở điểm (x2, y2). Thời gian tối thiểu mà Bob cần để đến và giải cứu cuộc sống của anh ta là bao nhiêu

- Dữ liệu vào:

Dòng đầu tiên chứa một số nguyên T biểu thị số lượng trường hợp thử nghiệm. Các dòng T sau đây mô tả các trường hợp thử nghiệm.

Dòng đầu tiên và duy nhất của mỗi trường hợp thử nghiệm chứa 6 số nguyên được phân tách bằng dấu cách: x1, y1, x2, y2, v1, v2

- Dữ liệu ra:

Xuất thời gian tối thiểu cần thiết với chính xác 5 chữ số sau dấu thập phân.

\*Ràng buộc:

T <= 100

-109 <= x1, y1, x2, y2, v1, v2<= 109

y1 <0vày2, v1, v2 > 0

## **Ý tưởng và thuật toán:**

Vận tốc của Bob trên bờ và trên biển là khác nhau, cho nên thời gian Bob cứu nạn nhân sẽ phụ thuộc vào vị trí Bob chọn để xuống biển(y=0), trên đoạn Oy, sẽ tồn tại một điểm mà tại đó, thời gian Bob di chuyển từ vị trí hiện tại đến nạn nhân là ngắn nhất nhất. Vậy thời gian mà Bob di chuyển tới chỗ nạn nhân là một hàm f() lõm dựa trên tọa độ x.

🡺Thỏa điều kiện sử dụng **Ternary Search.**

Thõa điều kiện với **Ternary search,** ta sẽ tam phân trên đoạn [x1,x2] với y =0

Ta sẽ tam phân trên đoạn y = 0 (trục Ox) với khoảng cách là x1(Bob) đến x2(người gặp nạn) với hàm xử lý đạt được kết quả nhỏ nhất để thõa mãn yêu cầu bài toán. Sau khi đã chặt được cái phân đoạn, ở hàm xử lý sẽ tính khoảng cách từ Bob đến điểm chặt và từ điểm chặt đến người gặp nạn. Với mỗi quãng đường tương ứng ta sẽ chia cho vận tốc chạy và vận tốc bơi để tính được thời gian Bob cứu người gặp nạn. Và tiếp tục ở mỗi phân đoạn khác với 100 lần tam phân thì sẽ cho ra kết quả bài toán

Thuật toán:

B1: Gán L= x1 và R=x2 và tìm vị trí mid1 và mid2 qua công thức mid1=L+(R-L)/3, mid=R-(R-L)/3

B2: Tính F(mid1) và F(mid2) với hàm xử lý F(d):

Trả về

(sqrt(pow(fabs(x1-d),2)+pow(y1,2)))/v1 + (sqrt(pow(fabs(d x2),2)+pow(y2,2)))/v2;

B3: So sánh F(mid1) và F(mid2) nếu:

F(mid1) > F(mid2) L= mid1 ngược lại R= mid2

B4: Lặp các bước trên 100 lần sẽ ra kết quả bài toán

# Bài 3: Chasing the Cheetah

Một đoàn làm phim dự định quay một đàn báo đang chạy. Họ định sẽ cho mỗi con báo vào một cái lồng hẹn giờ đặt cạnh nhau, sau một khoảng thời gian cửa lồng sẽ tự mở và con báo sẽ chạy. Được biết những con báo sẽ chạy song song nhau. Bạn được cung cấp thời gian bắt đầu và tốc độ của mỗi con báo. Chiều dài của đàn báo, đó là được định nghĩa là khoảng cách giữa con báo đầu tiên và con báo cuối cùng trong đàn, có thể khác nhau tại những thời điểm khác nhau. Tìm chiều dài tối thiểu của đàn báo trong quá trình chạy, trong đó tất cả các con báo phải chạy. Tất cả các lồng bắt đầu sẽ gần đến mức bạn có thể coi chúng ở cùng một chỗ. Con báo thứ K sẽ thả từ cái lồng bắt đầu của nó tại thời điểm Tk nhất định. Mỗi con báo sẽ chạy với 1 vận tốc Vk không đổi.

- Dữ liệu vào:

Có nhiều trường hợp thử nghiệm. Mỗi trường hợp chiếm nhiều dòng. Dòng đầu tiên của một trường hợp chứa số lượng con báo N (1 ≤N ≤ 100 000).

Tiếp theo, có N dòng, mỗi dòng chứa hai số nguyên Tk, VK cách nhau bởi khoảng trắng và biểu thị thời gian bắt đầu và vận tốc con báo Kth (1≤ K ≤ N). Tất cả các giá trị đầu vào Tk và Vk đều dương và nhỏ hơn 105. Chương trình kết thúc khi nhập số 0

- Dữ liệu ra:

Đối với mỗi trường hợp thử nghiệm, hãy in một dòng với một số dấu phẩy động L chỉ định mức tối thiểu chiều dài của lồng với độ chính xác sau dấu phẩy 2 số. Chiều dài của bầy là khoảng cách giữa con báo đầu tiên và con báo cuối cùng trong lồng. Độ dài có thể được đo bất cứ lúc nào T ≥ max (TK, K = 1, ..., N). Chúng tôi cho rằng mỗi con báo đang chạy với tốc độ không đổi trong suốt thời gian từ lúc bắt đầu.

## **Ý tưởng và thuật toán:**

Với mỗi thời gian t tại thì các con báo sẽ có 1 vị trí nhất định và ta sẽ tại thời điểm t đó thì ta cũng sẽ có chiều dài của đàn báo, với mỗi t thì chiều dài của đàn báo khác nhau, nhiệm vụ của chúng ta tìm ra chiều dài nhỏ nhất của đàn báo thông qua hàm xử lý f() với f() là hàm lõm. Như vậy ta đủ điệu kiện để sử dụng **Ternary search** trong khoảng thời gian con báo xuất phát cuối cùng đến 10^9

Tại mỗi thời điểm t ta sẽ có vị trí của mỗi con báo trong đàn (S=Vi \*(t-T)) tại mỗi thời điểm t, ta tìm ra quãng đường của con báo lớn nhất và nhỏ nhất, sau đó tính trị tuyết đối hiệu của 2 giá trị ấy sẽ ra được chiều dài của đàn báo. Và tiếp tục ở mỗi phân đoạn khác với 100 lần tam phân thì sẽ cho ra kết quả bài toán

Thuật toán:

B1: Gán L= 0 và R=10^9 và tìm vị trí mid1 và mid2 qua công thức mid1=L+(R-L)/3, mid=R-(R-L)/3

B2: Tính F(mid1) và F(mid2) với hàm xử lý F(t):

Duyệt qua mỗi con báo trong mảng từ i=0->n

(Si(T) = Vi\*(t-T)

Tìm max và min của Si của mỗi con báo, và cuối cùng trả về max-min

B3: So sánh F(mid1) và F(mid2) nếu:

F(mid1) > F(mid2) L= mid1 ngược lại R= mid2

B4: Lặp các bước trên 100 lần sẽ ra kết quả bài toán

# Bài 4: Devu và anh trai

Devu và anh trai yêu nhau rất nhiều. Vì họ là siêu chuyên viên máy tính, họ chỉ thích chơi với mảng. Họ được cha của họ cho hai mảng a và b. Mảng a được trao cho Devu và b cho anh trai mình. Như Devu thực sự là một đứa trẻ nghịch ngợm, anh ta muốn giá trị nhỏ nhất mảng a của mình có ít nhất càng nhiều càng giá trị lớn nhất mảng b của anh trai

Bây giờ bạn phải giúp Devu trong việc đạt được điều kiện này. Bạn có thể thực hiện nhiều thao tác trên các mảng. Trong một thao tác, bạn được phép giảm hoặc tăng bất kỳ phần tử nào của bất kỳ mảng nào bằng 1. Lưu ý rằng bạn được phép áp dụng thao tác trên bất kỳ chỉ mục nào của mảng nhiều lần. Bạn cần tìm số lượng yêu cầu tối thiểu cần thiết để đáp ứng điều kiện của Devu để anh em có thể chơi hòa bình mà không xích mích.

- Dữ liệu vào:

Dòng đầu chứa 2 số nguyên n,m (1 ≤ n, m ≤ 10^5)

Dòng tiếp theo chứa các phần tử mảng n và dòng cuối cùng chứa các phần tử mảng m

- Dữ liệu ra:

Đối với mỗi trường hợp thử nghiệm, hãy in một dòng với một số dấu phẩy động L chỉ định mức tối thiểu chiều dài của lồng với độ chính xác sau dấu phẩy 2 số. Chiều dài của bầy là khoảng cách giữa con vật đầu tiên và con vật cuối cùng trong lồng. Độ dài có thể được đo bất cứ lúc nào T ≥ max (TK, K = 1, ..., N). Chúng tôi cho rằng mỗi con báo đang chạy với tốc độ không đổi trong suốt thời gian từ lúc bắt đầu.

## **Ý tưởng và thuật toán:**

Gọi T là số bước thay đổi mảng a và b, x là giá trị tiêu chuẩn thõa mãn yêu cầu đề bài. Khi x quá lớn so với giá trị nhỏ nhất của mảng b thì số bước thay đổi trong mảng sẽ lớn và x nhỏ hơn hoặc tiệm cận giá trị nhỏ nhất so với giá trị nhỏ nhất của mảng b thì số bước thay đổi trong mảng sẽ nhỏ, nhiệm vụ của ta là tìm giá trị x sao cho số bước dịch chuyển trong cả 2 mảng a và b là nhỏ nhất với hàm f() lõm. Như vậy ta đủ điệu kiện để sử dụng **Ternary search**

Ta sẽ tam phân giá trị x từ 0 đến 10^9 với hàm xử lý đạt được kết quả nhỏ nhất để thõa mãn yêu cầu bài toán. Ở hàm xử lý, số bước sẽ tăng:

nếu x > a[i] thì số bước sẽ tăng lên x-a[i]

nếu b[i]> x thì số bước sẽ tăng lên b[i]-x

Và tiếp tục ở mỗi phân đoạn khác với 100 lần tam phân thì sẽ cho ra kết quả bài toán

Thuật toán:

B1: Gán L= 0 và R=10^9 và tìm vị trí mid1 và mid2 qua công thức mid1=L+(R-L)/3, mid=R-(R-L)/3

B2: Tính F(mid1) và F(mid2) với hàm xử lý F(x):

Ketqua = 0;

Duyệt qua mỗi phần tử trong mảng từ i=0->n

Nếu x> a[i] Ketqua+= x-a[i]

Duyệt qua mỗi phần tử trong mảng từ i=0->m

Nếu b > b[i] Ketqua+= b[i]-x

Trả về Ketqua

B3: So sánh F(mid1) và F(mid2) nếu:

F(mid1) > F(mid2) L= mid1 ngược lại R= mid2

B4: Lặp các bước trên 100 lần sẽ ra kết quả bài toán

# Bài 5: Poorness and Weakness

Bạn được cho 1 chuỗi N số nguyên a1, a2, a3, …, an Xác định số thực X mà điểm yếu của chuỗi a1-X, a2-X, …, an-X là nhỏ nhất có thể. Điểm yếu của chuỗi được xác định bởi giá trị lớn nhất của sự nghèo nàn trên mọi phân khúc Sự nghèo nàn trong một phân khúc được xác định bằng trị tuyệt đối tổng các phần tử của đoạn

- Dữ liệu vào:

Dòng đầu tiên chứa một số nguyên n (1 ≤  n  200000), độ dài của một chuỗi.

Dòng thứ hai chứa n số nguyên a1, a2, ..., an (| ai | ≤ 10000).

- Dữ liệu ra:

Một số thực biểu thị điểm yếu mức tối thiểu có thể của a1-x, a2-x, …., an-x, với độ chính xác sau dấu phẩy gồm 6 chữ số thập phân

## **Ý tưởng và thuật toán:**

Gọi X là điểm yếu của chuỗi, nhiệm vụ của ta sẽ tìm X sao cho chuỗi âm hoặc chuỗi dương là nhỏ nhất với mảng sau khi biến đổi là a1-X, a2-X, …, an-X với hãm f() lõm. Như vậy ta đủ điệu kiện để sử dụng **Ternary search**

Ta sẽ tam phân giá trị x từ 10^-4 đến 10^4 với hàm xử lý đạt được kết quả nhỏ nhất để thõa mãn yêu cầu bài toán, Ở hàm xử lý, ta duyệt các phần tử trong mảng để tính tổng chuỗi bằng a[i] -x sau đó tìm giá trị lớn nhất và nhỏ nhất của tổng ở mỗi lần duyệt qua 1 phần tử trong mảng. Điểm yếu của chuỗi sẽ bằng trị tuyệt đối hiệu của giá trị nhỏ nhất và lớn nhất. Và tiếp tục ở mỗi phân đoạn khác với 100 lần tam phân thì sẽ cho ra kết quả bài toán

Thuật toán:

B1: Gán L= 10^-4 và R=10^4 và tìm vị trí mid1 và mid2 qua công thức mid1=L+(R-L)/3, mid=R-(R-L)/3

B2: Tính F(mid1) và F(mid2) với hàm xử lý F(t):

Tong = 0;

Duyệt qua mỗi phần tử trong mảng từ i=0->n

Tong += a[i] -x

Tìm max và min của tổng sau môi lần duyệt qua 1 phần tử và cuối cùng trả về max-min

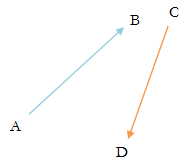
B3: So sánh F(mid1) và F(mid2) nếu:

F(mid1) > F(mid2) L= mid1 ngược lại R= mid2

B4: Lặp các bước trên 100 lần sẽ ra kết quả bài toán

# Bài 6: Closest Distance

Hai người đàn ông đang di chuyển đồng thời, một người đàn ông đang di chuyển từ A đến B và người đàn ông khác đang di chuyển từ C đến D. Ban đầu người đàn ông đầu tiên ở A, và người đàn ông thứ hai ở C. Họ duy trì vận tốc không đổi như vậy khi người đàn ông thứ nhất đạt đến B, cùng lúc người đàn ông thứ hai đến D. Bạn có thể giả sử rằng A, B, C và D là tọa độ 2D. Bạn phải tìm khoảng cách tối thiểu giữa họ dọc theo đường đi của họ.



**Input**

Bắt đầu bằng số nguyên T (1000), biểu thị số lượng trường hợp kiểm tra. Mỗi trường hợp sẽ chứa tám số nguyên: Ax, Ay, Bx, By, Cx, Cy, Dx, Dy. Tất cả các tọa độ nằm trong khoảng từ 0 đến 100. (Ax, Ay) biểu thị A. (Bx, By) biểu thị B và cứ thế.

**Output**

Đối với mỗi trường hợp, in số trường hợp và khoảng cách tối thiểu giữa chúng dọc theo đường dẫn của chúng. Các lỗi nhỏ hơn 10-6 sẽ bị bỏ qua.

## **Ý tưởng và thuật toán:**

Trên đường di chuyển của mình dĩ nhiên sẽ có lúc 2 người tiến gần nhau nhất. Nhiệm vụ chỉ là tìm ra thời gian để khoảng cách giữa hai người là nhỏ nhất. Vì hai người xuất phát và tới đích cùng 1 lúc, nên gọi thời gian lúc bắt đầu là 0 và khi tới đích là 1.

🡺Dùng **Ternary search** trên đoạn thời gian [0,1] để tìm ra khoảng cách ngắn nhất giữa 2 người.

Thuật toán:

B1: Nhập vào số lượng testcase T. Với mỗi testcase nhập vào tọa độ của 4 điểm A, B, C, D.

B2: Hàm xử lí f(t) có tham số truyền vào là thời gian t. Hàm trả về khoảng cách giữa 2 người vào lúc t.

B3: Đặt l=0 và r=1, mid1= l + (r – l)/3 và mid2= r – (r – l)/3. Nếu f(mid1)>f(mid2) thì l= mid1, ngược lại r = mid2. Chặt như vậy 100 lần thì ra được kết quả có sai số <10-6 .

# Bài 7: Những con kiến đáng ghét

Nhà Thanh có nhiều kiến, Thanh định diệt kiến bằng cách căng một sợi dây bọc đường ra giữa nhà, không ngoài dự kiến, sau một lúc kiến bu đầy trên dây.

Sau khi quan sát thì Thanhđã tính được có tất cả **N** con kiến trên sợi dây, nếu ta xem sợi dây là trục **Ox**, nút trái của sợi dây là **–oo**, nút phải sợi dây là **+oo** thì con thứ **i** hiện tại đang ở tọa độ **a[i]**. Mỗi con kiến đang bò theo 1 hướng với một vận tốc cố định **v[i]**. **v[i]** là số dương nếu con kiến đang bò từ trái sang phải với vận tốc là **v[i]** (đơn vị/s), **v[i]** là số âm nếu con kiến đang bò từ phải sang trái với vận tốc là **–v[i]**. Như vậy tại thời điểm **t** nào đó thì con kiến thứ **i** sẽ có vị trí là **p[i] = a[i] + t\*v[i]**.Thanh chỉ có thể dùng chai thuốc xịt kiến để xịt một lần lên một vị trí nào đó trên sợi dây để giết chết toàn bộ các con kiến một lượt (vì nếu sau lần đầu xịt mà còn các con kiến còn sống thì chúng sẽ phát hiện ra mai phục và chạy trốn ngay). Giả sử Thanh xịt thuốc lên điểm có tọa độ **x** với liều lượng thuốc là **E** vào một thời điểm nào đó thì các con kiến có tọa độ nằm trong đoạn **[x – E; x + E]** sẽ bị tiêu diệt, nếu một thời điểm nào đó cả **N** con kiến đều đến cùng một vị trí thì ta chỉ cần xịt lên đúng 1 điểm đó với một lượng thuốc rất nhỏ, trường hợp này ta có thể xem như **E = 0** (Xem ví dụ để hiểu rõ hơn). Do chai thuốc xịt kiến này là đồ mượn nên Thanh cũng muốn xài tiết kiệm…

**Yêu cầu:**Hãy giúp Thanhtính xem liều lượng thuốc tối thiểu **E** mà anh ta cần dùng để tiêu diệt được toàn bộ lũ kiến, biết rằng Thanhcó thể xịt thuốc ngay lập tức hoặc quyết định đợi một khoảng thời gian thích hợp để xịt thuốc tiêu diệt toàn bộ lũ kiến.

**Input:**

* Dòng đầu tiên chứa số **N** là số lượng các con kiến.
* Dòng thứ 2 chứa **N** số nguyên **a[i]** (i từ 1 đến N) là vị trí hiện thời của **N** con kiến
* Dòng thứ 3 chứa **N** số nguyên **v[i]** (i từ 1 đến N) là vận tốc của **N** con kiến.
* **-109  ≤ a[i], v[i] ≤ 109**

**Output:**

* Một số thực duy nhất là giá trị liều lượng **E** nhỏ nhất mà **Thanh** cần sử dụng. In ra kết quả làm tròn đến **3** chữ số thập phân.

## **Ý tưởng và thuật toán:**

Đầu tiên hãy nhận xét: Với khoảng thời gian ban đầu các con kiến đi về hai hướng khác nhau sẽ làm co lại khoảng cách, đến một mức nào đó khoảng cách lại bị dãn ra đến cực đại.

→ Đây là một hàm f() lõm, vấn đề ta là tìm min sao cho nhanh trên dãy này.

→ Dùng ternary search trên khoảng thời gian chờ với số lần chặt là 100, với mỗi khoảng thời gian T ta xem xét E = ((a[i]+v[i]\*T) max - (a[i]+v[i]\*T) min)/2.

→ Tìm được E nhỏ nhất là kết quả bài toán.

Thuật toán:

B1: Nhập vào số lượng kiến N, vị trí và vận tốc của mỗi con kiến.

B2: Hàm xử lí f(t) có tham số truyền vào là thời gian t. Hàm trả về khoảng cách của con kiến nằm bên trái ngoài cùng và con kiến bên phải ngoài cùng vào thời gian t. Bằng cách tính tất cả vị trí của những con kiến, rồi so sánh để tìm ra kết quả.

B3: Dùng Ternary Search trên đoạn thời gian [0,105]. Đặt l=0 và r=105, mid1= l + (r – l)/3 và mid2= r – (r – l)/3. Nếu f(mid1)>f(mid2) thì l= mid1, ngược lại r = mid2. Chặt như vậy 100 lần thì ra được kết quả có sai số <10-3

# Bài 8: Building Construction

Cho N tòa nhà có chiều cao h1, h2, h3 ... hn, mục tiêu là làm cho mọi tòa nhà có chiều cao bằng nhau. Điều này có thể được thực hiện bằng cách loại bỏ các viên gạch khỏi tòa nhà hoặc thêm một số viên gạch vào tòa nhà. Việc loại bỏ một viên gạch hoặc thêm một viên gạch được thực hiện với một chi phí nhất định, ứng với mỗi tòa nhà. Tìm chi phí tối thiểu mà bạn có thể làm cho các tòa nhà trông đẹp, bằng cách xây dựng lại các tòa nhà sao cho các tòa nhà N thỏa mãn h1 = h2 = h3 = .. = hn = k (k có thể là bất kỳ số nào).

Input: Dòng đầu tiên chứa số nguyên T biểu thị số lượng trường hợp thử nghiệm. Điều này sẽ được theo sau bởi 3 dòng T, 3 dòng trên mỗi trường hợp thử nghiệm. Dòng đầu tiên của mỗi trường hợp thử nghiệm chứa một số nguyên n và dòng thứ hai chứa n số nguyên biểu thị chiều cao của các tòa nhà [h1, h2, h3 .... hn] và dòng thứ ba chứa n số nguyên [c1, c2, c3 ... cn] biểu thị chi phí thêm hoặc xóa một đơn vị gạch khỏi tòa nhà tương ứng. T <= 15; n <= 10000; 0 <= Hi <= 10000; 0 <= Ci <= 10000

Output: Đầu ra phải chứa T dòng mỗi dòng tương ứng với một testcase.

## **Ý tưởng và thuật toán:**

Hãy gọi h là **chiều cao lí tưởng**, nghĩa là tất cả tòa nhà đều sẽ được chỉnh sửa cho cao bằng h, ở đây 0<=h<=hmax, hmax là chiều cao của tòa nhà cao nhất. Nhiệm vụ của ta là tìm ra h để chi phí chỉnh sửa các toàn nhà là ít nhất, nghĩa là các tòa nhà sẽ ít bị chỉnh sửa nhất.

Khi h thay đổi thì chi phí để chỉnh sửa các tòa nhà thay đổi, ban đầu, h nhỏ thì ta sẽ phải chỉnh sửa các tòa nhà khá nhiều, nên chi phí chỉnh sửa sẽ cao. Khi tăng h lên thì chi phí sẽ giảm, vì các tòa nhà sẽ bị chỉnh sửa ít hơn, cứ như vậy chi phí giảm đến một mức cực tiểu, nếu tiếp tục tăng h thì các tòa nhà sẽ buộc phải chỉnh sửa nhiều hơn cho bằng với **chiều cao lí tưởng h**, chi phí sẽ tăng trở lại.

🡺Chi phí chỉnh sửa các toàn nhà là một hàm f() lõm. Vấn đề là tìm min sao cho nhanh trên dãy này.

🡺Dùng **Ternary search** trên khoảng [0,hmax] để tìm ra cực tiểu của hàm.

Thuật toán:

B1: Nhập T là số lượng testcase, nhập n là số lượng tòa nhà, nhập chiều cao và chi phí của từng tòa nhà.

B2: Hàm xử lí f(x): hàm có 1 tham số truyền vào là chiều cao lí tưởng x, với mỗi x truyền vào, hàm sẽ trả về chi phí cần thiết để chỉnh sửa các tòa nhà, bằng cách duyệt toàn bộ chiều cao và chi phí của từng tòa nhà, và tính ra tổng chi phí.

B3: Dùng Ternary search Đặt l=0 và r=hmax, mid1= l + (r – l)/3 và mid2= r – (r – l)/3. Nếu f(mid1)>f(mid2) thì l= mid1, ngược lại r = mid2. Cứ như vậy đến khi nào r – l < 3 thì dừng, sau cùng chỉ cần trả về giá trị nhỏ nhất của f() trong khoảng [l,r] lúc này đã nhỏ hơn 3.

# Bài 9: Trick or treat

Johnny và những người bạn của mình đã quyết định dành đêm Halloween để thực hiện bộ sưu tập kẹo thông thường từ các hộ gia đình trong làng của họ. Vì ngôi làng quá lớn để một nhóm thu thập kẹo từ tất cả các ngôi nhà một cách tuần tự, Johnny và bạn bè đã quyết định tách ra để mỗi người đến một ngôi nhà khác nhau, thu thập kẹo (hoặc tàn phá nếu cư dân không phát kẹo) và quay lại điểm hẹn được sắp xếp trước. Có n ngôi nhà trong làng, vị trí của chúng có thể được xác định bằng tọa độ Đề-các của chúng trên mặt phẳng Ơ-clit. Băng đảng Johnny cũng được tạo thành từ n người (bao gồm cả chính Johnny). Họ đã quyết định phân phát kẹo sau khi mọi người quay lại với chiến lợi phẩm của họ. Những ngôi nhà có thể ở rất xa, nhưng sở thích của Johnny là ăn kẹo càng sớm càng tốt. Bọn trẻ sẽ gặp nhau ở con sông (Những điểm có tung độ y=0), chúng có thể di chuyển dọc theo bất kỳ hướng nào trên mặt đất. Đúng nửa đêm, mỗi đứa trẻ sẽ gõ cửa căn nhà mà nó đã chọn, thu thập kẹo ngay lập tức và đi bộ dọc theo con đường ngắn nhất đến điểm hẹn. Nói với Johnny vào thời gian nào cả nhóm sẽ gặp lại.

**Input:** Dòng đầu chứa số nguyên N chỉ số nhà (1 ≤ n ≤ 50 000). N dòng tiếp theo mô tả vị trí của các ngôi nhà; mỗi dòng này chứa hai số thực x và y (−200 000<= x, y<= 200 000), tọa độ của một ngôi nhà tính bằng mét. Các ngôi nhà ở các vị trí khác nhau. Một dòng trống sau mỗi trường hợp. Một dòng có n = 0 chỉ ra kết thúc của đầu vào; không viết bất kỳ đầu ra cho trường hợp này.

**Output:** Đối với mỗi trường hợp thử nghiệm, in hai số trong một dòng cách nhau bởi một khoảng trắng: tọa độ x của điểm gặp gỡ trên dòng y = 0, sao cho thời gian đứa trẻ cuối cùng đến điểm hẹn là tối thiểu và thời gian nó đến (tính bằng giây sau nửa đêm) . Câu trả lời của bạn phải chính xác trong phạm vi sai số tuyệt đối hoặc tương đối 10-5.

## **Ý tưởng và thuật toán:**

Cả nhóm Johnny sẽ gặp nhau ở con sông (đường thẳng **y=0** trên mặt phẳng tọa độ). Nhiệm vụ đưa ra là tìm ra điểm tập hợp của cả nhóm sao cho thời gian để cả nhóm tụ hội là tối thiểu.

Gọi **(x,0)** là tọa độ mà cả nhóm sẽ tập hợp, xmin<=x<=xmax, với xmin là hoành độ nhỏ nhất của một căn nhà trong danh sách, và xmax là hoành độ lớn nhất. Tất nhiên sẽ tồn tại một điểm (x,0) mà khoảng cách từ nó đến tất cả ngôi nhà là nhỏ nhất.

🡺 Chỉ cần dùng **Ternary search** để tìm ra x trên đoạn [xmin, xmax]

Thuật toán:

B1: Nhập số căn nhà n, nhập tọa độ các căn nhà, so sánh tọa độ để tìm ra xmin và xmax.

B2: Hàm xử lí f(x) có tham số truyền vào là hoành độ x, hàm sẽ trả về trả về khoảng cách từ điểm (x,0) đến tòa nhà xa nhất, bằng cách duyệt qua tọa độ của tất cả căn nhà và so sánh.

B3: Đặt l=xmin và r=xmax, mid1= l + (r – l)/3 và mid2= r – (r – l)/3. Nếu f(mid1)>f(mid2) thì l= mid1, ngược lại r = mid2. Chặt như vậy 100 lần thì ra được kết quả có sai số <10-5

# Bài 10: Elections

Bạn sắp tranh cử lãnh đạo của một thành phố nhỏ và muốn thắng cử.Bạn có danh sách n người sẽ đi bầu cử. Mỗi người sẽ gồm 2 chỉ số: ứng viên mà người đó sẽ bầu (a[i], a[i] = 0 là bầu cho bạn) và số tiền bạn cần hối lộ để người đó bầu mình (b[i], b[i] = 0 nếu a[i] = 0). Tuy nhiên, bạn muốn tối thiểu hóa số tiền phải chi nhưng vẫn đảm bảo thắng cử.

Yêu cầu: Cho danh sách người bầu cử, in ra chi phí cần thiết.

Input:

● Dòng đầu số nguyên n (n ≤ 10^5).

● Dòng i trong n dòng sau chứa 2 số nguyên a[i] và b[i] (0≤ a[i] ≤10^5, 0 ≤ b[i] ≤ 10^4).

Output:

● In ra số tiền bạn cần chi để thắng cử.

## **Ý tưởng và thuật toán:**

Hãy định nghĩa **“mức ứng cử”** là số người tối thiểu bầu cho mình và không có ứng cử viên nào vượt qua mức đó được - ký hiệu là T ( 1 ≤ T ≤ n).

+ Nếu ta chọn mức ứng cử là T, thì những ứng viên có số người bầu cho mình >= T bạn chắc chắn phải mua chuộc 1 số trong đó để số người theo các ứng viên phải đảm bảo <T (mới thắng được)

+ Để cho chi phí tối thiểu, dĩ nhiên bạn sẽ chọn những người mà b[i] nhỏ nhất mà theo các ứng viên bạn buộc phải mua chuộc để cho đạt điều kiện <T là được.

+ Nếu số người bạn mua chuộc vẫn chưa đủ >= T bạn buộc phải mua thêm 1 lượng người cho đủ, gọi x là số người bạn đã mua chuộc, bạn cần mua thêm (T-x) người nữa, và cũng để tối ưu bạn lấy (T-x) người b[i] nhỏ nhất trong số các người mà bạn chưa mua chuộc.

+ Duyệt hết các giá trị T, ta sẽ có đáp án

- Với ***mức ứng cử*** nhỏ, hầu như ta phải mua chuộc tất cả các người bầu cử, khi ta tăng dần ***mức ứng cử*** thì số người phải bầu cử ta giảm → số tiền phải mua chuộc giảm.

- Sau khi giảm đến cực tiểu, số tiền lại bị tăng lên do ta phải ***lấp đầy mức ứng cử***, tức là các ứng viên khác có lẽ đã thua phiếu nhưng ta phải đảm bảo đủ số người bầu theo yêu cầu thuật toán.

→ Hàm T() thỏa điều kiện và ta có thể áp dụng ***Ternary Search Min*** trên hàm này

Thuật toán:

B1: Nhập vào số người bầu cử n, nhập a[i] tượng trưng cho ứng viên, b[i] là giá tiền mua chuộc của từng người. Với mỗi ứng viên, sắp xếp những người bầu cho họ theo thứ tự số tiền mua chuộc phải bỏ ra.

B2: Hàm xử lí f(x) có 1 tham số gọi là **Mức ứng cử**, hàm trả về tổng chi phí cần bỏ ra nếu chọn mức ứng cử như vậy. Hàm sẽ duyệt qua từng ứng viên, ứng viên nào có số người bầu >= **Mức ứng cử** thì đếm số người cần mua chuộc để < **Mức ứng cử,** đồng thời ghi lại những người chưa bị mua chuộc. Sau khi duyệt qua tất cả ứng viên, nếu số người bầu cho bản thân vẫn chưa bằng **Mức ứng cử**, thì mua chuộc thêm những người còn lại cho đủ.

B3: Sử dụng Ternary Search, đặt l=0 và r=105, mid1= l + (r – l)/3 và mid2= r – (r – l)/3. Nếu f(mid1)>f(mid2) thì l= mid1, ngược lại r = mid2. Cứ như vậy đến khi nào r – l < 3 thì dừng, sau cùng chỉ cần trả về giá trị nhỏ nhất của f() trong khoảng [l,r] lúc này đã nhỏ hơn 3.