CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM

Độc lập – Tự do – Hạnh phúc

MÔ TẢ SÁNG KIẾN

Mã số: .....

1. Tên sáng kiến:

NÂNG CAO PHƯƠNG PHÁP DUYỆT BẰNG THUẬT TOÁN TÌM KIẾM NHỊ PHÂN

2. Lĩnh vực áp dụng sáng kiến: Bồi dưỡng học sinh giỏi Tin học

3. Mô tả bản chất của sáng kiến:

3.1.Tình trạng giải pháp đã biết:

Duyệt là một trong những phương pháp được sử dụng để giải quyết các bài toán trong tin học, đặc biệt là các bài toán trong các kì thi học sinh giỏi. Cụ thể là ở các bài toán tìm kiếm. Thực tế cho thấy trong những năm gần đây qua các kì thi học sinh giỏi cấp khu vực, quốc gia,…Yêu cầu của đề thi không những học sinh phải tìm được kết quả chính xác ở mỗi bài toán mà đòi hỏi thuật toán đặt ra giải quyết bài toán phải thực hiện tối ưu về thời gian.

Việc tìm được kết quả bài toán chính xác các em đã đạt được. Tuy nhiên còn về thời gian thực hiện thuật toán thì rất ít các em đạt được yêu cầu này dẫn đến các em không được điểm tối đa chỉ ở mức 30% điểm hoặc 50% điểm của mỗi bài toán.

Tìm kiếm nhị phân là một trong những thuật toán góp phần cải thiện được thời gian khi thực hiện chương trình cho các bài toán tìm kiếm.

Việc vận dụng thuật toán này cho các bài toán tìm kiếm như thế nào nhằm giúp các em khắc phục được tình trạng thời gian thực hiện chương trình. Đây là giải pháp mà trong quá trình nghiên cứu, giảng dạy, bản thân tôi đã áp dụng trong năm học vừa qua.

3.2 Nội dung giải pháp đề nghị công nhận là sáng kiến:

Mục đích của giải pháp:

Nâng cao hiệu quả thực hiện chương trình, tối ưu thời gian thực hiện thuật toán, kết hợp tìm kiếm nhị phân với thuật toán khác để giải hiệu quả một vài bài toán tìm kiếm với lượng dữ liệu lớn trong các kì thi học sinh giỏi.

Trong phạm vi đề tài này tôi vận dụng thuật toán tìm kiếm nhị phân để hướng dẫn các em giải quyết một vài bài toán dạng tìm kiếm trong công tác giảng dạy các lớp chuyên tin và bồi dưỡng học sinh giỏi, thay vì các em vận dụng phương pháp duyệt khác thì không giải quyết được trọn vẹn yêu cầu đề bài

hoặc các em chưa biết cách vận dụng kết hợp thuật toán này với các thuật toán khác để giải quyết bài toán.

Cũng qua đề tài, tôi muốn cùng các anh, chị đồng nghiệp trao đổi, trao dồi nhằm góp phần nâng cao trình độ chuyên môn trong công tác giảng dạy và bồi dưỡng học sinh giỏi bộ môn.

Nội dung giải pháp.

Tư tưởng thuật toán tìm kiếm nhị phân:

Tìm kiếm nhị phân là thuật toán tìm kiếm phần tử (khóa) trong một dãy cho trước mà các phần tử trong dãy đã được sắp xếp. So với các thuật toán tìm kiếm khác tìm kiếm nhị phân có ưu điểm là thu hẹp nhanh phạm vi tìm kiếm sau mỗi lần so sánh khóa với phần tử được chọn.

Đầu tiên chọn phần tử ở "giữa dãy" để so sánh với khóa, trong đó giữa được xác định như sau: giữa = (đầu + cuối) div 2

Khi đó chỉ xảy ra một trong ba trường hợp sau đây:

- Nếu phần tử giữa dãy =khóa thì giữa là chỉ số cần tìm rồi kết thúc.
- Nếu phần tử giữa dãy > khóa thì việc tìm kiếm tiếp theo chỉ xét trên dãy $a_{đầu}, a_{đầu+1}, \ldots, a_{giua\ -1}$ (phạm vi tìm kiếm mới bằng một nửa phạm vi tìm kiếm trước đó).
- Nếu phần tử giữa dãy < khóa thì việc tìm kiếm tiếp theo chỉ xét trên dãy $a_{giua+1}, a_{giua+2}, \ldots, a_{cuối}$.

Quá trình trên sẽ được lặp lại một số lần cho đến khi hoặc đã tìm thấy khóa trong dãy hoặc không tìm thấy.

Thuật toán này có độ phức tạp là: O(LogN).

Tóm tắt thuật toán như sau:

Tử tưởng này được thể hiện qua các bài toán sau đây:

Bài toán mở đầu: Dãy con tăng dài nhất

Cho một dãy số gồm N số nguyên $a_1, a_2, \ldots, a_N$. Hãy tìm độ dài dãy con tăng dài nhất trong dãy trên.

Dữ liệu vào: Daycon.inp

- Dòng đầu tiên số nguyên dương N
- Dòng thứ hai gồm N số mỗi số cách nhau bằng một dấu cách

Dữ liệu ra: Daycon.out

- Một số duy nhất độ dài dãy con tìm được

Giới hạn: $1 \le N \le 10^6$, $0 < a_1, a_2, \ldots, a_N \le 10^9$

Ví dụ:

Daycon.inp	**Daycon.out**
7 9 4 5 6 4 5 7	4

Nhận xét:

Nhìn vào bài toán thì ta có thể tìm kết quả ngay bằng phương pháp quy hoạch động như sau:

Gọi F[i] là độ dài của dãy con tăng dài nhất kết thúc tại a[i]

Cơ sở quy hoạch động: F[0] = 1

Công thức quy hoạch động:

$$F[i] = Max\ \{F[j]\} với\ 0\ \le j \le i\ và\ a[j] \le < a[i]$$

Đánh giá độ phức tạp sau khi thực hiện thuật toán là $O(N^2)$

Tuy nhiên theo yêu cầu đề bài $1 \le N \le 10^6$. Nên ta dùng phương pháp quy hoạch động là không hiệu quả.

Ta tiếp cận bài toán theo hướng tìm kiếm nhị phân như sau:

F[i] là mảng chứa vị trí của số hạng nhỏ nhất của các dãy con tăng có độ dài là i.

Tại mỗi bước, nếu A[i] lớn hơn phần tử lớn nhất trong dãy con tăng dài nhất hiện thời thì bổ sung A[i] vào cuối dãy ngược lại ta dùng tìm kiếm nhị phân tìm kiếm vị trí thích hợp để đặt A[i] vào nhằm đảm bảo rằng tại mỗi vị trí sẽ có một phần tử nhỏ nhất là đuôi của dãy con tăng có độ dài i.

Code chương trình

```
#include <iostream>
#include <fstream>
using namespace std;
ifstream fi ("daycon.inp");
ofstream fo ("daycon.out");
long int n, a[1000000],f[1000000],kq,d,c,g;
void nhap()
{
    fi>>n;
```

```
	for (int i=1;i<=n;i++)
		fi>>a[i];
	fi.close();
}
void xet()
{
	f[1]=1; kq=1;
	for (int i=2;i<=n;i++)
		if (a[i] > a[f[kq]])
		{
		kq++;
		f[kq]=i;}
		else if (a[i] < a[f[kq]])
		{
		d=1;c=kq;
		while (d<c)
		{
		g=(d+c)/2;
		if (a[f[g]]< a[i])
			d=g+1;
		else c=g-1;}
		f[d]=i; }
}
int main()
{
	nhap();
	xet();
	fo<<kq;
	fo.close();
}
```

Bài toán 1: Cắt đoạn

Cho n đoạn dây điện ($1 \le n \le 10^5$). Đoạn dây thứ i có độ dài a_i ($0 < a_i \le 10^9$). Cần phải cắt các đoạn dây đã cho thành các đoạn sao cho có được K đoạn dây bằng nhau có độ dài nguyên. Có thể không cần cắt hết các đoạn dây đã cho. Mỗi đoạn dây bị cắt có thể có phần còn thừa khác 0.

Yêu cầu: Xác định độ dài lớn nhất của đoạn dây có thể nhận được. Nếu không có cách cắt thì đưa ra số 0.

Dữ liệu vào: Catdoan.inp

- Dòng đầu chứa hai số nguyên N, K
- Dòng thứ i trong N dòng tiếp theo chứa số nguyên a_i

Dữ liệu ra: Catdoan.out

- Một số duy nhất là độ dài lớn nhất có thể nhận được.

Ví dụ:

Catdoan.inp	Catdoan.out
4 11 802 743 547 539	200

Thuật toán:

- Đầu tiên ta tính tổng độ dài n đoạn dây

- Ta nhị phân với dau=1,cuoi=tong/k

+ Giua=(dau+cuoi)/2

+ Ta kiểm tra giua. Nếu đúng thì cập nhật kết quả là giua và d=g+1 để mong tìm độ dài đoạn lớn hơn giua đó.

Hàm kiểm tra:

+ Ta duyệt từ 1 cho tới n.

+ Dem+=a[i]/g (ta tính với đoạn dây thứ i ta có thể chia ra bao nhiêu đoạn dây có độ dài là g)

+ Nếu dem>=k thì return true ngược lại thì false

Code chương trình

```
#include<bits/stdc++.h>
using namespace std;
ifstream fi("catdoan.inp");
ofstream fo("catdoan.out");
```

```
int a[100001],n,k,tb;
void doc()
{
  fi>>n>>k;
  for(int i=1;i<=n;i++)
  {
        fi>>a[i];
        tb+=a[i];
  }
  tb=tb/k;
  fi.close();
}
bool kiemtra(int x)
{
    int dem=0;
    for(int i=1;i<=n;i++)
         dem+=a[i]/x;
    if(dem>=k) return true;
    return false;
}
void  nhiphan()
{
    int res=0;
    int d=1,c=tb,g;
    while (d<=c)
    {
              g=(d+c)/2;
                if(kiemtra(g))
        {
           res=g;
           d=g+1;
        }
```

```
        else c=g-1;
    }
    fo<<res;
    fo.close();
}
int main()
{
    doc();
    nhiphan();
}
```

Bài toán 2: KÉN RỂ **Tên chương trình Kenre.***

Phú Ông có một bè gỗ liêm. Để việc vận chuyển dễ dàng và tránh nhầm lẫn khi chọn gỗ để làm nhà, gỗ trên bè được cắt ra thành N đoạn, ở hai đầu của mỗi đoạn gỗ được ghi chỉ số đầu là a và chỉ số cuối là b (a < b). Để cưới được con gái phú ông Bòm phải chọn ra được nhiều đoạn gỗ nhất trong N đoạn trên bè kèm theo yêu cầu của Phú ông như sau: Đầu tiên lấy ra một đoạn bất kì có chỉ số đầu là a và chỉ số cuối đoạn là b, sau đó thực hiện lấy các đoạn khác sao cho: chỉ số a và b của đoạn lấy ra sau chứa đoạn vừa được lấy trước đó ($a_{sau} <= a_{trước}$, $b_{sau} >= b_{trước}$. Bòm tiếp tục lấy cho đến khi không tìm được đoạn thỏa mãn yêu cầu nữa. Em hãy lập trình để giúp Bòm lấy được nhiều đoạn gỗ nhất có thể lấy.

Dữ liệu vào: Kenre.inp

- Dòng đầu tiên chứa số nguyên dương N, là số đoạn gỗ liêm có trên bè
- Dòng thứ i trong N dòng tiếp theo chứa 2 số nguyên a và b là chỉ số đầu và chỉ số cuối cho đoạn gỗ thứ i.

Dữ liệu ra: Kenre.out

Một số duy nhất là kết quả của bài toán

Giới hạn: $1 \le N \le 10^6$, $1 \le a < b \le 10^6$

Ví dụ:

Kenre.inp	**Kenre.out**	**Kenre.inp**	**Kenre.out**
3 1 2 5 6 2 9	2	7 7 9 2 6 1 7 3 4	4

		1 5 2 5 1 4	

Nhận xét:

- Theo yêu cầu của bài toán thì test ví dụ 1 chọn nhiều nhất được 2 đoạn là: 2-9 và 5-6.
- Theo yêu cầu của bài toán thì test ví dụ 2 chọn nhiều nhất được 4 đoạn là: 1-7, 2-6, 2-5, 3-4.

Thuật toán:

Dùng mảng a để lưu chỉ số đầu của đoạn và mảng b lưu chỉ số cuối của đoạn

B1: Sắp xếp giảm dần theo chỉ số đầu a, nếu có trường hợp a[i] =a[i+1] thì ta sắp tăng theo chỉ số cuối của b.

B2: Dùng thuật toán tìm kiếm nhị phân tìm độ dài dãy con không giảm dài nhất của mảng b. Độ dài này chính là kết quả cần tìm của bài toán.

Code chương trình

```
#include <bits/stdc++.h>
using namespace std;
ifstream fi ("Kenre.inp");
ofstream fo ("Kenre.out");
int n;
int a[1000001], b[1000001], f[1000001];
void nhap()
{
   fi>>n;
   for (int i=1; i<=n; i++)
      fi>>a[i]>>b[i];
   fi.close();
}
int qs (int l, int r)
{
   int i=l; int j=r;
   int k=l;
   while (i<=j)
```

```
    {
        while ((a[i]>a[k]) || ((a[i]==a[k])&&(b[i]<b[k])))
            i++;
        while ((a[j]<a[k]) || ((a[j]==a[k])&&(b[j]>b[k])))
            j--;
        if (i<=j)
        {
            swap (a[i],a[j]);
            swap(b[i],b[j]);
            i++;
            j--;
        }
    }
    if (l<j)
        qs (l,j);
    if (i<r)
        qs(i,r);
}
void xuly()
{
    f[1]=1; int kq=1;
    for (int i=2; i<=n; i++)
    {
        if (b[i]>=b[f[kq]])
        {
            kq++;
            f[kq]=i;
        }
        if (b[i]<b[f[kq]])
        {
                int d=1, c=kq;
                while (d<c)
```

```
            {
               int g=(d+c)/2;
               if (b[f[g]]<b[i])
                  d=g+1;
               else
                  c=g-1;
            }
           f[d]=i;
      }
   }
   fo <<kq; fo.close();
}
int main()
{
   nhap();
   qs(1,n);
   xuly();
}
```

Bài toán 3 : BÚP BÊ GỖ **Tên file chương trình Bupbe.***

Cửa hàng đồ chơi nhập về n con búp bê gỗ. Các con búp bê được đánh số từ 1 tới n trong đó con búp bê thứ i là một hộp rỗng có kích thước là một số nguyên a_i. Người ta có thể lồng con búp bê thứ i vào trong con búp bê thứ j nếu con búp bê thứ j đang rỗng và $a_i+k \le a_j$, với k là một số nguyên dương cho trước. Bằng cách lồng các con búp bê vào nhau theo cách như vậy, cửa hàng chỉ cần tìm chỗ đặt những con búp bê ngoài cùng (những con búp bê không nằm trong bất kỳ con búp bê nào khác) vào kho.

Yêu cầu: Hãy giúp cửa hàng lồng các con búp bê vào nhau sao cho tổng kích thước các con búp bê ngoài cùng là ***nhỏ nhất***.

Dữ liệu vào: Bupbe.inp gồm 2 dòng

- Dòng 1 chứa hai số nguyên dương $n \le 10^5$; $k \le 10^9$ cách nhau một khoảng trắng.

- Dòng 2 chứa n số nguyên dương $a_1, a_2, ..., a_n$ ($a_i \le 10^9$), mỗi số cách nhau một khoảng trắng.

Dữ liệu ra: Bupbe.out

- Là một số nguyên duy nhất là tổng kích thước các con búp bê ngoài cùng theo phương án tìm được.

Ví dụ:

Bupbe.inp	Bupbe.out
8 2 8 4 2 1 1 3 5 9	18

Tư tưởng: Tìm và Loại bỏ những phần tử lớn nhất có giá trị nhỏ hơn hoặc bằng a[i]-k. (dãy đã sx giảm)

Thuật toán:

- Sắp xếp dãy giảm dần
- Duyệt từ đầu dãy: 1 ->n. Lần lượt tìm những phần tử lớn nhất có giá trị nhỏ hơn hoặc bằng a[i]-k và chưa đánh dấu(d[i]=true). nếu thỏa thì đánh dấu phần tử tại vị trí trong dãy là false
- Duyệt lại từ đầu dãy tìm kết quả bài toán :
  Kq:=0;
  Nếu d[i]=true thì kq:=kq+a[i]

Code chương trình

```
#include <fstream>
using namespace std;
ifstream fi ("Bupbe.Inp");
ofstream fo ("Bupbe.Out");
int n;bool dd [1000000];
long k,a [100000],kq=0;
void nhap()
{
   fi >> n >> k;
   for (int i=0;i<n;i++)
      fi >> a[i];
   fi.close();
}
void init()
```

```
{
   for (int i=0;i<=n;i++)
      dd[i]=true;
}
void quicksort(int d,int c)
{
   int i=d,j=c,x=a[(d+c)/2];
   while (i<=j)
   {
      while (a[i]>x)  i++;
      while (a[j]<x) j--;
      if (i<=j)
      {
         if (i<j)    swap(a[i],a[j]);
         i++;j--;
      }
   }
   if (d<j)    quicksort(d,j);
   if (i<c)    quicksort(i,c);
}
void tim(int x)
{
   int dau=0,cuoi=n-1,tam=-1;
   while (dau<=cuoi)
   {
      int giua=(dau+cuoi)/2;
      if (a[x]>=a[giua]+k && dd[giua])   {   cuoi=giua-1;tam=giua;    }
      else    dau=giua+1;
   }
   if (tam!=-1)
      dd[tam]=false;
}
```

```
int main()
{
   nhap();quicksort(0,n-1);init();
   for (int i=0;i<n;i++)
      tim(i);
   for (int i=0;i<n;i++)
   if (dd[i])  kq+=a[i];
   fo << kq;
   fo.close();
}
```

Bài toán 4: XEM TI VI **Tên file chương trình XemTV.***

Có n chương trình giải trí, chương trình thứ i ($1 \le i \le n$) có thời điểm bắt đầu là s_i và thời điểm kết thúc là t_i. Chương trình giải trí thứ i và chương trình giải trí thứ j (với $1 \le i < j \le n$) được gọi là không phù hợp với nhau về lịch phát sóng nếu người xem không thể xem trọn vẹn nội dung của cả hai chương trình giải trí này. Nếu thời điểm kết thúc t_i của chương trình i là thời điểm bắt đầu s_j của chương trình j thì hai chương trình này vẫn được xem là có lịch phát sóng phù hợp với nhau.

Ví dụ: Có 3 chương trình giải trí như sau: Chương trình 1 (s_1= 7, t_1= 10), chương trình 2 (s_2= 12, t_2= 15), chương trình 3 (s_3= 10, t_3= 20). Chương trình 1 và chương trình 2 có lịch phát sóng phù hợp với nhau. Tương tự, chương trình 1 và chương trình 3 cũng được xem là có lịch phát sóng phù hợp với nhau. Tuy nhiên, chương trình 2 và chương trình 3 có lịch phát sóng ***không*** phù hợp với nhau.

Yêu cầu: Cho biết kế hoạch phát sóng của N chương trình giải trí, hãy xác định có bao nhiêu cặp chương trình có lịch phát sóng không phù hợp với nhau.

Dữ liệu nhập: file **Xemtv.inp** gồm các dòng sau:

- Dòng đầu tiên chứa một số nguyên dương n (với $n \le$ **50.000**).

- Dòng thứ i trong số n dòng tiếp theo ($1 \le i \le n$), mỗi dòng gồm hai số nguyên dương s_i và t_i là thời điểm bắt đầu và thời điểm kết thúc của chương trình giải trí thứ i (với $1 \le s_i < t_i \le 10^5$). Các số trên cùng một dòng được ghi cách nhau bởi 1 khoảng trắng.

Dữ liệu xuất: file **Xemtv.out**

- Là một số nguyên xác định số lượng cặp chương trình có lịch phát sóng ***không phù hợp*** với nhau.

Ví dụ:

Xemtv.inp	Xemtv.out	Xemtv.inp	Xemtv.out
3 7 10 12 15 10 20	1	3 1 3 3 5 5 7	0

Thuật toán:

- Sắp xếp mảng s tăng dần (mảng t theo s)
- Ta xét i từ 1-> n:

  + Với mỗi i ta thực hiện tìm kiếm nhị phân tìm giá trị s[g] lớn nhất (điều kiện:s[g]<t[i])

  + Dem+=g-i;
- Kết quả bài toán là dem

Code chương trình

```
#include<bits/stdc++.h>
using namespace std;
ifstream fi("xemtv.inp");
ofstream fo("xemtv.out");
int n,s[50001],t[50001];

void doc()
{
   fi>>n;
   for(int i=1;i<=n;i++)
      fi>>s[i]>>t[i];
   fi.close();
}

void quicksort(int l,int r)
{
   if(l>=r) return;
   else
```

```
    {
        int check=s[(l+r)/2];
        int i=l,j=r;
        while(i<=j)
        {
            while(s[i]<check) i++;
            while(s[j]>check) j--;
            if(i<=j)
            {
                swap(t[i],t[j]);
                swap(s[i],s[j]);
                i++;
                j--;
            }
        }
        quicksort(l,j);
        quicksort(i,r);
    }
}

int nhiphan(int i)
{
    int res=i;
    int d=i+1,c=n,g;
    while(d<=c)
    {
        g=(d+c)/2;
        if(s[g]<t[i])
        {
            res=g;
            d=g+1;
        }
```

```
        else c=g-1;
    }
    return res;
}
int main()
{
long  dem=0;
    doc();
    quicksort(1,n);
    for(int i=1;i<=n;i++)
    dem+=nhiphan(i)-i;
    fo<<dem;
    fo.close();
}
```

Bài toán 5: THI NẤU ĂN **Tên file chương trình Nauan.***

Có n bạn sinh viên đang tham gia dự thi nấu ăn nhân dịp năm mới và được đánh số báo danh từ 1 đến n, bạn sinh viên thứ i tham dự với số lượng là a_i món ăn. Ban tổ chức sẽ đánh số các món ăn dự thi như sau: các món ăn của thí sinh thứ nhất đánh số từ 1 đến a_1, các món ăn của thí sinh thứ hai đánh số từ a_1+1 đến a_1+a_2.... và tương tự như vậy cho đến món cuối cùng. Sau khi chấm thi, Ban tổ chức chọn trao giải cho m món ăn với các số hiệu là p_1, p_2, ..., p_m. Hãy cho biết các món ăn đạt giải đó thuộc về các bạn sinh viên nào?

Dữ liệu vào: Nauan.inp gồm 4 dòng

- Dòng thứ nhất là số nguyên n ($1 \le n \le 10^5$) là số thí sinh tham gia dự thi.
- Dòng thứ hai là n số nguyên a_1, a_2, ..., a_n ($1 \le a_i \le 10^4$) là số lượng món ăn của từng thí sinh, mỗi số cách nhau một khoảng trắng.
- Dòng thứ ba là số nguyên m ($1 \le m \le 10^4$) là số lượng món ăn đạt giải.
- Dòng thứ tư là m số nguyên p_1, p_2, ..., p_m là số hiệu của m món ăn đạt giải, mỗi số cách nhau một khoảng trắng.

Dữ liệu ra: Nauan.out

- Là m số nguyên s_1, s_2, ..., s_m cho biết số báo danh thí sinh của từng món ăn đạt giải (món ăn p_i là của thí sinh số báo danh s_i), mỗi số cách nhau một khoảng trắng.

Ví dụ:

Nauan.inp	Nauan.out
5 5 4 1 2 3 3 5 6 12	1 2 4

Thuật toán:

- Ta sẽ tạo mảng tong[i] với ý nghĩa tổng các phần tử mảng a từ 1 đến i
- Duyệt với từng p[i] thì ta thực hiện tìm kiếm nhị phân:
    + Ta tìm tong[g] nào lớn hơn hoặc bằng p[i] (g nhỏ nhất).
    + Xuất g

Code chương trình

```
#include<bits/stdc++.h>
using namespace std;
ifstream fi("nauan.inp");
ofstream fo("nauan.out");
int n,a[10001],m,p[1001];
long tong[10001];

void doc()
{
   fi>>n;
   for(int i=1;i<=n;i++)
   fi>>a[i];
   fi>>m;
   for(int i=1;i<=m;i++)
   fi>>p[i];
   fi.close();
}

void taomang()
```

```
{
   tong[0]=0;
   for(int i=1;i<=n;i++)
   tong[i]=tong[i-1]+a[i];
}

int tknp(int x)
{
   int res=-1;
   int d=1,c=n,g;
   while(d<=c)
   {
      g=(d+c)/2;
      if(tong[g]>=x)
         {
            res=g;
            c=g-1;
         }
       else d=g+1;
   }
   return res;
}

void xuly()
{
   for(int i=1;i<=m;i++)
   fo<<tknp(p[i])<<" ";
   fo.close();
}

int main()
{
```

```
    doc();
    taomang();
    xuly();
}
```

Bài toán 6: Picnic **Tên chương trình là Picnic.***

Nhân dịp lễ 30/4, Dinh cùng các bạn của mình tổ chức một cuộc vui chơi ở khu vui chơi giải trí giống như đợt đi trại hè vừa rồi. Nhắc lại bài toán: Nhà của Dinh và các bạn của em nằm trên cùng 1 con đường, các nhà được đánh vị trí từ 1 đến N, mỗi nhà cách nhau 1 mét. Nhà của Dinh ở vị trí **1** và khu vui chơi ở vị trí **N**. Nhà **M** người bạn của Dinh ở các vị trí $\mathbf{a_1, a_2,..., a_M}$. Ngoài ra trên tuyến đường còn có **P** trạm xe buýt tại các vị trí $\mathbf{b_1, b_2, ..., b_P}$. Từ nhà mình, Dinh sẽ đi đến nhà của các bạn. Bạn ấy có thể đi bằng taxi hoặc xe buýt. Với taxi, Dinh có thể bắt từ bất kì vị trí nào, giá của taxi là **T đồng/mét**. Với xe buýt, Dinh chỉ có thể bắt từ trạm này và đi đến một trạm khác, giá của xe buýt là **B đồng/lượt** không phân biệt khoảng cách. Tuy rằng trong túi đang rủng rỉnh tiền thưởng học sinh giỏi cấp quốc gia vừa qua nhưng Dinh vẫn phải tìm cách đi **tiết kiệm tiền nhất** sao cho thăm đủ tất cả các bạn và đến khu giải trí N để chơi được nhiều trò chơi hơn. Sau khi nghe kể về chuyến trại hè thầy Sơn đã gợi ý Dinh rằng em nên thăm nhà các bạn **theo thứ tự bất kì** để tiết kiệm được nhiều tiền nhất. Bạn hãy giúp Dinh tìm cách đi đón tất cả các bạn và đến điểm vui chơi với số tiền phải trả là ít nhất nhé!

Yêu cầu: Cho biết số nhà trên đường, các nhà phải đến đón, số trạm xe buýt và số tiền đi xe taxi, xe buýt, bạn hãy tìm cách đi sao cho đến **thăm tất cả các nhà** và đến vị trí N **với số tiền ít nhất**.

Dữ liệu vào : Picnic.inp

- Dòng thứ nhất chứa các số nguyên N, M, P, T, B là số nhà, các nhà phải đón, số trạm xe buýt và số tiền đi taxi, xe buýt. ($1 \le N \le 10^9 \mid 0 \le M \le 20 \mid 0 \le P \le 1000 \mid 1 \le T,B \le 10^4$).

- Dòng thứ hai chứa M số nguyên là các nhà phải đến, số thứ a_i là vị trí của nhà thứ i ($1 \le a_i \le N$). Dữ liệu cho đảm bảo không có 2 nhà trùng vị trí.

- Dòng cuối cùng chứa P số nguyên là vị trí các trạm xe buýt theo thứ tự tăng dần, số thứ b_i là vị trí của trạm thứ i, mặc định có trạm ở vị trí 1 và N. ($1 \le b_i \le N$).

Dữ liệu ra: Picnic.out

- Dòng thứ nhất chứa 1 số nguyên duy nhất là số tiền ít nhất phải trả.

- Dòng thứ hai chứa M số nguyên là thứ tự Dinh thăm nhà các bạn của mình. Nếu có nhiều đáp án, chỉ cần in ra 1 cách đi bất kì.

Ví dụ:

Picnic.inp	Picnic.out
10 2 2 1000 2000 5 8 4 7	8000 1 2

Nhận xét:

- Đầu tiên Dinh đi xe buýt từ 1 đến 4
- Sau đó đi taxi từ 4 đến 5, 5 đến 8 và 8 đến 10

Tổng số tiền là 2000+1000+3000+2000=8000 đồng

Thuật toán:

- Sort vị trí các căn nhà từ nhỏ đến lớn(thứ tự nhà không đổi), đồng thời sort vị trí các bến xe bus
- Với mỗi vị trí căn nhà thứ i áp dụng tìm kiếm nhị phân tìm kiếm xebus gần vị trí nhà thứ i nhất. Thì giá tiền di chuyển giữa 2 vị trí có thể lựa chọn đi toàn bộ bằng xe taxi hoặc đi 1 đoạn = xe bus +taxi.

Code chương trình

```
#include <bits/stdc++.h>
using namespace std;
typedef pair<long,long> ll;
ll a[10001];
long long n,Bs[10001],m,p,t,b;
#define d first
#define tt second
void Docfile()
{
   cin >> n >> m >> p >> t >> b;
   for (int i=1;i<=m;i++)
   {
      cin >> a[i].d;
      a[i].tt=i;
```

```
    }
    sort(a+1,a+1+m);
    a[0].d=1;
    a[m+1].d=n;
    for (int i=1;i<=p;i++)
        cin >> Bs[i];
    Bs[0]=1;
    Bs[p+1]=n;
    sort(Bs+1,Bs+p+1);
}
int Nearest(long long x)
{
    long long dau=0,cuoi=p+1,mid,kq=-1;
    while (dau<=cuoi)
    {
        mid=(dau+cuoi)/2;
        if (Bs[mid]==x)
            return mid;
        if (Bs[mid]>x)
            cuoi=mid-1;
        else
        {
            kq=mid;
            dau=mid+1;
        }
    }
    if (x-Bs[kq]<=Bs[kq+1]-x)
        return kq;
    return kq+1;
}

void Xuli()
```

```
{
   long long res=0;
   for (int i=1;i<=m+1;i++)
   {
      int neara=Nearest(a[i].d),nearb=Nearest(a[i-1].d);
      res+=min(abs(a[i].d-a[i-1].d)*t,abs(Bs[neara]-a[i].d)*t+abs(Bs[nearb]-a[i-
1].d)*t+b);
   }
   cout << res << endl;
   for (int i=1;i<=m;i++)
      cout << a[i].tt << " ";
}
int main()
{
   freopen("picnic.inp","r",stdin);
   freopen("picnic.out","w",stdout);
   Docfile();
   Xuli();
}
```

3.3 Khả năng áp dụng của giải pháp

Sau khi áp dụng giải pháp này các em trong đội tuyển và các em ở lớp chuyên tin rất tự tin và chắc chắn khi gặp các bài toán ở dạng tìm kiếm với lượng dữ liệu cho rất lớn là các em vận dụng ngay hoặc kết hợp tìm kiếm nhị phân với các thuật toán khác để giải tìm được kết quả bài toán thay vì các phương pháp trước đây vẫn tìm được kết quả nhưng còn hạn chế về thời gian thực hiện thuật toán.

Bằng những bài toán từ đơn giản đến phức tạp qua việc phân tích và vận dụng hiệu quả thuật toán tìm kiếm nhị phân. Giải pháp này không chỉ áp dụng cho các em ở các lớp chuyên tin mà còn góp phần không ít vào công tác bồi dưỡng học sinh giỏi cấp tỉnh cho các em học sinh trung học phổ thông trong tỉnh.

3.4. Hiệu quả, lợi ích thu được hoặc dự kiến có thể thu được do áp dụng giải pháp.

Hiệu quả của việc vận dụng thuật toán làm cho các em học sinh mới bắt đầu học chuyên tin đam mê, tự tin hơn qua việc rút ngắn phạm vi tìm kiếm so

với các phương pháp duyệt khác bởi tìm kiếm nhị phân có độ phức tạp tối ưu là O(logN)

Thể hiện rõ nhất là ở kì thi Olympic 30/04 năm 2018 các em đạt 5/6 giải trong đó. Khối 10: 02 huy chương vàng, 01 huy chương bạc, khối 11: 01 huy chương bạc và 01 huy chương đồng.

Kì thi trại hè phương nam cho các em học sinh khối 10 năm học 2017-2018: 01 huy chương vàng, 01 huy chương bạc.

3.5. Tài liệu kèm theo gồm

- Đĩa CD chứa Code +test của các bài tập đặt ra để giải quyết vấn đề.

Bến Tre, ngày 17 tháng 3 năm 2019