Arbore de intervale

Este un **AB** echilibrat(1) (diferența în valoare absolută a înălțimilor subarborilor stâng și drept este <=1), în care fiecărui nod i se asociază o structură suplimentară de informații (interval). Pentru un interval **[st,dr]** se poate asocia un **AI** notat **T(st, dr)**, construit recursiv, astfel:

• La nodul rădăcină se asociază informația asociată din **[st, dr]**;
• dacă **st < dr** atunci
- subarborelui stâng i se asociază **T(st, mij)**
- subarborelui drept **T(mij + 1, dr)**

* mij este mijlocul intervalului **[st, dr]**.

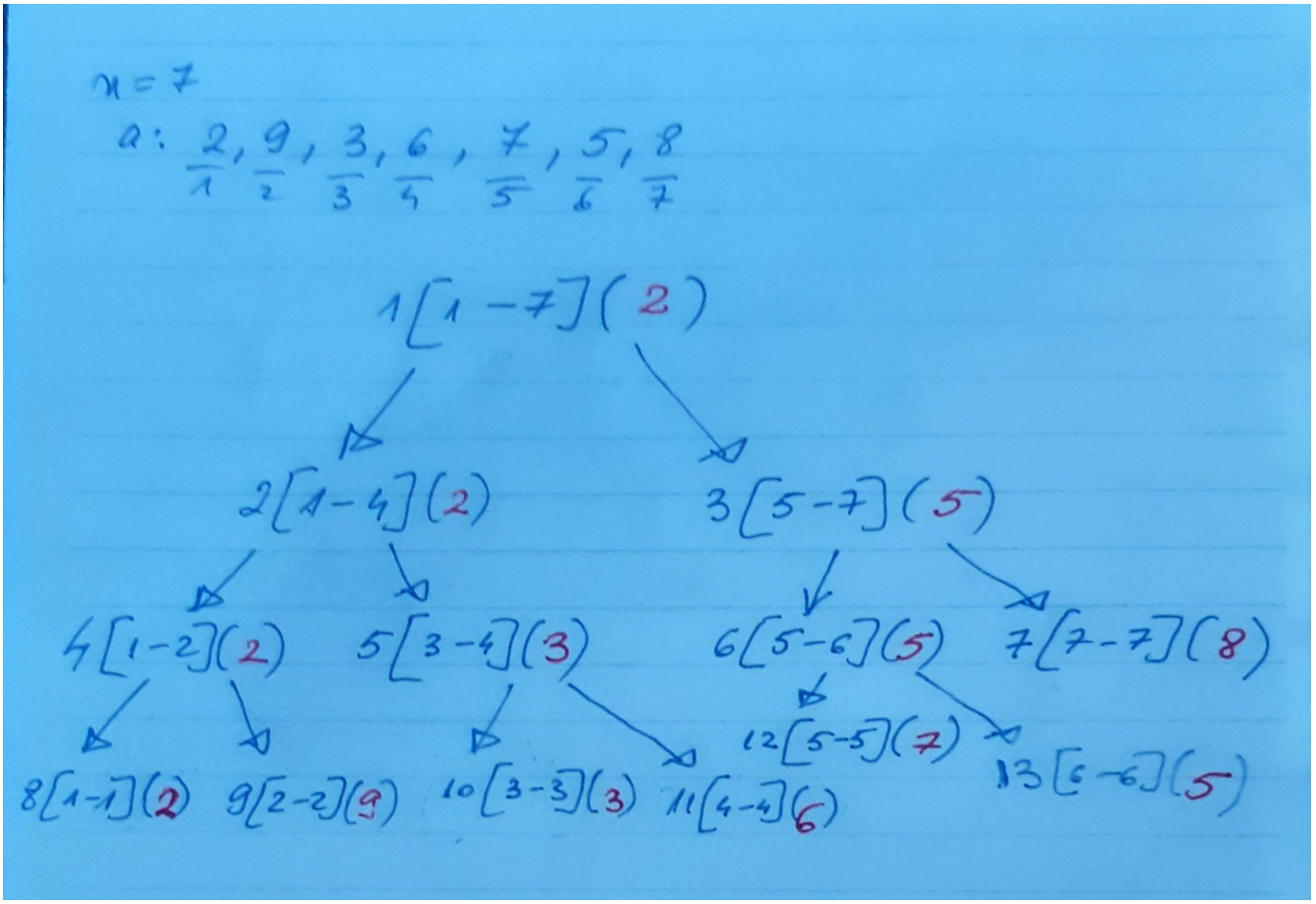
Intervalul asociat unui nod se numește interval standard.

(1) adâncimea unui arbore de intervale cu n intervale(noduri) este **log2 n + 1**

Pentru stocarea în memorie a unui **AI** se utilizează un vector v unde elementul de la poziția p memorează informația aferentă nodului p. Această informație a fost obținută prin prelucrarea informațiilor (conform criteriului de prelucrare) din cei doi fii stânga și dreapta ai nodului **p** (Acești fii au informațiile asociate lor memorate în v la pozițiile **2*p** și **2*p+1**);

Exemplu:

Considerăm vectorul `a={2,9,3,6,7,5,8}` avînd `n=7` elemente. Dorim să determinăm valoarea minimă din vector.



Vectorul v asociat AI va avea **2*7-1** noduri=**13** noduri

```
v={2, 2, 5, 2, 3, 5, 8, 2, 9,  3,  6,  7,  5}
   1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13
```

A. Crearea AI

```
void Create(int x,int st,int dr)
{
    if(st==dr)
    {
        v[x]=a[st];
    }
    else
    {
        int m=(st+dr)/2;
        Create(2*x,st,m);
        Create(2*x+1,m+1,dr);
        v[x]=min(v[2*x],v[2*x+1]);
    }
}
```

B. Modificarea AI

```
void Replace(int x,int st, int dr, int p)
/// x- radacina AI, st-dr limitele asociate AI, p-pozitia elem ce a
fost modificat
{
    if(st==dr)
        v[x]=a[dr];
    else
    {
        int m=(st+dr)/2;
        if(p<=m)
            Replace(2*x,st,m,p);
        else
            Replace(2*x+1,m+1,dr,p);
        v[x]=minim(v[2*x],v[2*x+1]);
    }
}
```

C. Interogarea AI

```
int Query(int x, int st, int dr, int a, int b)
{
    if (a <= st && dr <= b)
        return v[x];
    else
        {
            int mij = (st + dr) / 2, q1,q2;
            if (a <= mij) q1= Query(2 * x, st, mij, a, b);
            if (b > mij) q2= Query(2 * x + 1, mij + 1, dr, a, b);
            if(q1<q2) return q1;
            else return q2;
        }
}
```

D. Implementarea

```
int main()
{
    int n,a[100],v[200],rad,st,dr;
    cin>>n;
    for(i=1; i<=n; i++)
        cin>>a[i];
    st=1,dr=n;rad=1;
    Create(rad,st,dr);
    for(i=1;i<2*n;i++)
        cout<<v[i]<<" ";
    cout<<endl;
    cin>>poz>>val;
    a[poz]=val;
    Replace(rad,st,dr,poz);
    for(i=1;i<=n;i++) cout<<a[i]<<" ";
    cout<<endl;
    for(i=1;i<2*n;i++)  cout<<v[i]<<" ";
    cout<<Query(1,1,7,4,6);
    return 0;
}
```