[ডাটা স্ট্রাকচার: সেগমেন্ট ট্রি-১](http://www.shafaetsplanet.com/planetcoding/?p=1557" \o "Permanent Link to ডাটা স্ট্রাকচার: সেগমেন্ট ট্রি-১)

লেখক: শাফায়েত তারিখ: জুনe ১১, ২০১৩ বিভাগ: [কনটেস্ট প্রোগ্রামিং/প্রবলেম সলভিং](http://www.shafaetsplanet.com/planetcoding/?cat=35), [প্রোগ্রামিং](http://www.shafaetsplanet.com/planetcoding/?cat=1)

ট্যাগ: [ডাটা স্ট্রাকচার](http://www.shafaetsplanet.com/planetcoding/?tag=%e0%a6%a1%e0%a6%be%e0%a6%9f%e0%a6%be-%e0%a6%b8%e0%a7%8d%e0%a6%9f%e0%a7%8d%e0%a6%b0%e0%a6%be%e0%a6%95%e0%a6%9a%e0%a6%be%e0%a6%b0)• [ডিভাইড এন্ড কনকোয়ার](http://www.shafaetsplanet.com/planetcoding/?tag=%e0%a6%a1%e0%a6%bf%e0%a6%ad%e0%a6%be%e0%a6%87%e0%a6%a1-%e0%a6%8f%e0%a6%a8%e0%a7%8d%e0%a6%a1-%e0%a6%95%e0%a6%a8%e0%a6%95%e0%a7%87%e0%a6%be%e0%a7%9f%e0%a6%be%e0%a6%b0)• [সেগমেন্ট ট্রি](http://www.shafaetsplanet.com/planetcoding/?tag=%e0%a6%b8%e0%a7%87%e0%a6%97%e0%a6%ae%e0%a7%87%e0%a6%a8%e0%a7%8d%e0%a6%9f-%e0%a6%9f%e0%a7%8d%e0%a6%b0%e0%a6%bf)

তুমি হয়তো এরকম প্রবলেম কনটেস্টে দেখেছ, একটি ইন্টিজার অ্যারে দেয়া আছে আর অনেকগুলো কুয়েরি দেয়া আছে। প্রতিটা কুয়েরিতে বলেছে একটা রেঞ্জের মধ্যে সবগুলো সংখ্যার যোগফল বলতে। অ্যারের সাইজ ১০^৫, কুয়েরির সংখ্যাও ১০^৫! বুঝতেই পারছো প্রতি কুয়েরিতে লুপ চালিয়ে যোগফল বের করতে পারবেনা। কিভাবে প্রবলেমটি সলভ করবে?

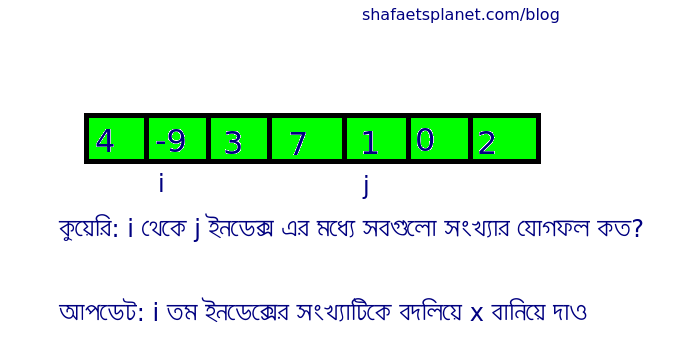
এটা সলিউশন খুব সহজ, তোমাকে কিউমুলেটিভ সাম রাখতে হবে। ধরো একটি অ্যারে আছে sum[MAX], তাহলে sum[i] তে রাখবে ১ থেকে i নম্বর ইনডেক্স পর্যন্ত সবগুলো সংখ্যার যোগফল। i থেকে j পর্যন্ত যোগফল বের করতে দিলে(i<=j) sum[j]-sum[i-1] হবে তোমার উত্তর। বুঝতে না পারলে নিচের উদাহরণটা দেখো:

ইনপুট:  
arr[]={4, -9, 3, 7, 1, 0, 2}

cumulative sum বের করবো:  
sum[0]=0;  
(for i=1;i<=n;i++) sum[i]=sum[i-1]+arr[i];

তাহলে cumulative sum হবে:  
sum[] ={4, -5, -2, 5, 6, 6, 8}

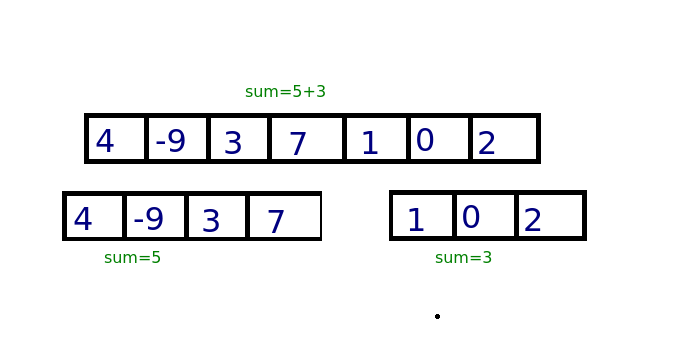
এটা একদম বাচ্চাদের কাজ, তুমি ৫মিনিটে কোড করে ফেলতে পারবে। কিন্তু প্রবলেমসেটার তোমাকে বিপদে ফেলতে বললো কুয়েরির করার মাঝে মাঝে অ্যারেটি বদলে দেয়া হবে!! মাঝে মাঝে তোমাকে বলবে i-তম ইনডেক্সের সংখ্যাটিকে x বানিয়ে দিতে, আবার আগের মতো যোগফলও বলতে বলবে। এখন কি করবে?

[](http://www.shafaetsplanet.com/planetcoding/wp-content/uploads/2013/06/seg1.png)

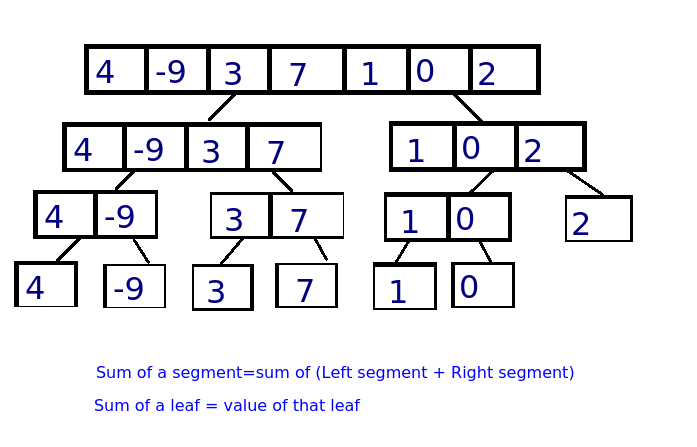
এবার আর কিউমুলেটিভ সাম দিয়ে কাজ হবেনা, তোমার দরকার হবে সেগমেন্ট ট্রি নামের অসাধারণ শক্তিশালী একটা ডাটা স্ট্রাকচার। ইউনিভার্সিটিতে ডাটা স্ট্রাকচার কোর্সে তোমাকে এটা পড়াবেনা, কিন্তু এটা ব্যবহার করে অনেক কাজ করা যায়।

পরের অংশে যাবার আগে তোমার কিছু জিনিস সম্পর্কে ধারণা পরিষ্কার থাকতে হবে। রিকার্শন সম্পর্কে কোনো রকম অস্পষ্টতা থাকলে আপাতত সামনে না আগানোই ভালো। এছাড়া তুমি যদি মার্জ সর্ট সম্পর্কে জানো তাহলে সেগমেন্ট্রি বোঝা তোমার জন্য খুব সহজ হয়ে যাবে। এছাড়া ট্রি সম্পর্কে যদি কিছুই না জানো তাহলে সেগমেন্ট ট্রি এখনই শেখা কি ঠিক হবে? ঠিক মার্জ সর্টের মতো সেগমেন্ট্রি ট্রিও “ডিভাইড এন্ড কনকোয়ার” পদ্ধতিতে কাজ করে।

ডিভাইড এন্ড কনকোয়ার পদ্ধতির মূল কথা হলো একটা প্রবলেমকে ভেঙে ছোটো ছোটো অংশ বানাও, আগে সেই ছোট অংশ সলভ করো এবং ছোটো অংশের সলিউশন থেকে বড় অংশের সলিউশন বের করো। আমরা তাই অ্যারেটাকে ২টা অংশে ভাগ করে ফেলবো এবং দুইটা ভাগের যোগফল আলাদা করে বের করবো।

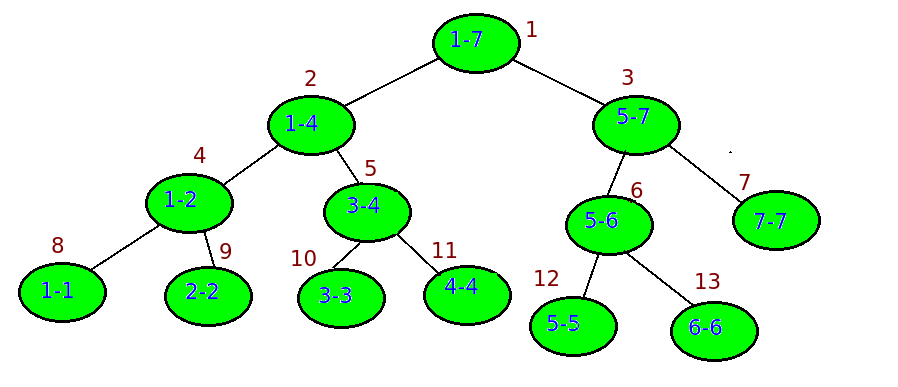
[](http://www.shafaetsplanet.com/planetcoding/wp-content/uploads/2013/06/seg2.png)

তুমি যদি বাম আর ডান পাশের ভাগের যোগফল আলাদা করে বের করতে পারো তাহলে খুব সহজেই বড় অংশটার যোগফল বের করতে পারবে। আমি বলার আগেই বুঝতে পারছো এরপরে কি করবো। ছোটো অ্যারেগুলোকে আরো টুকরা করবো যতক্ষণনা ১ সাইজের টুকরা পাই। ১ সাইজের টুকরোর যোগফল আমরা জানি, সেখান থেকে বড়গুলোর যোগফল বের করে ফেলবো।

[](http://www.shafaetsplanet.com/planetcoding/wp-content/uploads/2013/06/seg3.png)

ছবিটা দেখতে বিদঘুটে হলেও জিনিসটা খুবই সহজ। আমরা অ্যারেটাকে ভাঙতে ভাঙতে ছোট করে ফেলেছি, সবথেকে ছোট অংশের(লিফ নোড) যোগফল আমরা জানি, সেখান থেকে বড় গুলো সহজেই বের করতে পারবো বাম এবং ডানের অংশ যোগ করে।

ছবিটায় প্রতিটা সেগমেন্টকে যদি একটা নোড ধরি তাহলে একটা ট্রি তৈরি হয়ে গিয়েছে, প্রতিটা নোডে আছে একটা অংশ বা রেঞ্জের যোগফল। এটার নামই সেগমেন্ট ট্রি। এখন তোমার মনে হতে পারে এই জিনিস দিয়ে কিভাবে i থেকে j অংশের যোগফল বের করবে কারণ আমরাতো ভাঙছি সম্পূর্ণ অ্যারেটা আর সবশেষে পাচ্ছি সবটুকুর যোগফল। কিছুক্ষণের মধ্যে এটার উত্তর পাবে। আমরা ট্রি টাকে একটু অন্যভাবে দেখি:

[](http://www.shafaetsplanet.com/planetcoding/wp-content/uploads/2013/06/seg4.png)

খেয়াল করে দেখো আগের ট্রি টাই একেছি কিন্তু এবার সেগমেন্টগুলো পুরোটা না দেখিয়ে শুধু রেঞ্জটা লিখেছি। যেমন ৩ নম্বর নোডে আছে ৫ থেকে ৭ ইনডেক্সের সবগুলোর যোগফল। নোডের নাম্বারিং টা গুরুত্বপূর্ণ। রুট নোড হবে ১, তার বামের নোড হবে ১\*২=২, এবং ডানের নোড হবে (১\*২+১)=৩। অর্থাৎ রুট x হলে বামেরটা হবে 2x এবং ডানেরটা 2x\*1। বাইনারি ট্রি অ্যারেতে স্টোর করার জন্য সুবিধাজনক পদ্ধতি এটা।

এখন আগে দেখি কিভাবে এই স্ট্রাকচারটা তৈরি করবো। নিচের কোডটি দেখো, ব্যাখ্যা আছে কোডের নিচে:

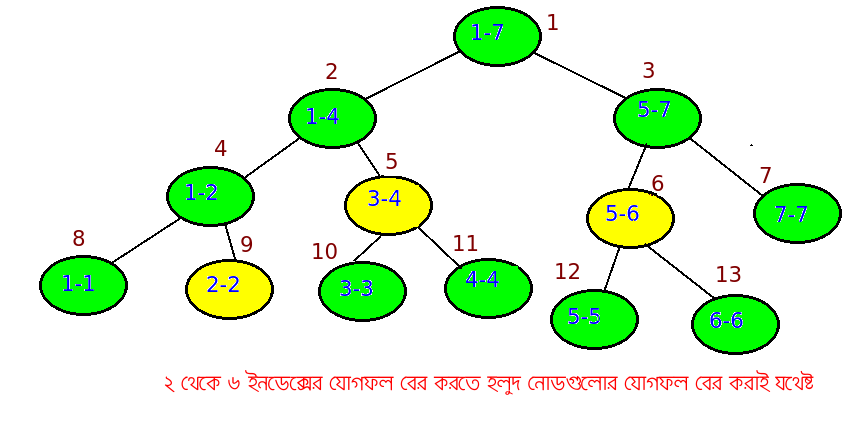


|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27 | #define mx 100001  int arr[mx];  int tree[mx\*3];  void init(int node,int b,int e)  {      if(b==e)      {          tree[node]=arr[b];          return;      }      int Left=node\*2;      int Right=node\*2+1;      int mid=(b+e)/2;      init(Left,b,mid);      init(Right,mid+1,e);      tree[node]=tree[Left]+tree[Right];  }  int main()  {      //READ("in");      int n;      cin>>n;      repl(i,n)cin>>arr[i];      init(1,1,n);        return 0;  } |

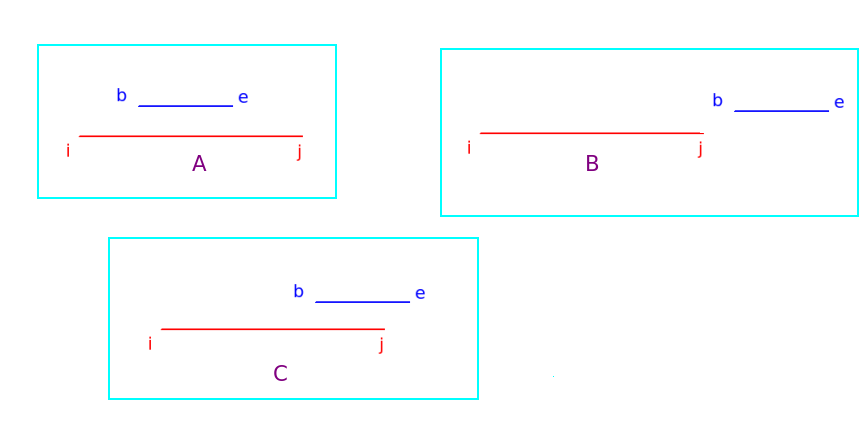
tree[] অ্যারেতে আমরা ট্রি টাকে স্টোর করবো। ট্রি অ্যারের সাইজ হবে ইনপুট অ্যারের ৩গুণ(কেন??)। init ফাংশনটি arr অ্যারে থেকে ট্রি তৈরি করে দিবে। init এর প্যারামিটার হলো node,b,e, এখানে node=বর্তমানে কোন নোডে আছি এবং b,e হলো বর্তমানে কোন রেঞ্জে আছি। শুরুতে আমরা নোড ১ এ থাকবো এবং ১-৭ রেঞ্জে থাকবো(ট্রি এর ছবিটা দেখো)।

যদি (b==e) হয়ে যায় তাহলে আমরা শেষ নোডে পৌছে গেছি, এখানে যোগফল হবে অ্যারেতে যে ভ্যালু আছে সেটাই, সেটাকে ট্রিতে স্টোর করে রিটার্ণ করে দিলাম। যদি (b==e) না হয় তাহলে অ্যারেটা কে দুই ভাগে ভাগ করতে হবে। বাম পাশের নোডের ইনডেক্স হবে node\*2 এবং ডান পাশেরটা node\*2+1। এবং অ্যারেটাকে ভাঙবো ঠিক মাঝখানে। এবার রিকার্সিভলি দুই পাশে init কল করলে বাম এবং ডান পাশের ছোটো অংশের যোগফল বের হয়ে যাবে। দুইপাশের কাজ শেষ হয়ে গেলে বর্তমান নোডের যোগফল হবে বাম এবং ডানের নোডের যোগফল।  
বুঝতে সমস্যা হলে কোডটা হাতে-কলমে একবার সিমুলেট করো, তাহলেই পরিষ্কার হয়ে যাবে।

এইবার আমাদের একটা কুয়েরি ফাংশন দরকার যেটা i থেকে j এর মধ্যে সবগুলো সংখ্যার যোগফল বলে দিবে। ধরো i=2 এবং j=6। তাহলে লক্ষ্য করো নিচের হলুদ রঙের নোডগুলোর যোগফলই তোমার উত্তর:

[](http://www.shafaetsplanet.com/planetcoding/wp-content/uploads/2013/06/seg5.png)

2-6 রেঞ্জের জন্য হলুদ নোডগুলো আমাদের রিলেভেন্ট নোড, বাকিগুলো এক্সট্রা। আমাদের কুয়েরি ফাংশনের কাজ হবে শুধু রিলেভেন্ট নোডগুলোর যোগফল বের করা। কোডটা init ফাংশনের মতোই হবে তবে কিছু কন্ডিশন অ্যাড করতে হবে। ধরো তুমি এমন একটা নোডে আছো যেখানে b-e রেঞ্জের যোগফল আছে। তুমি এই নোডটা রিলেভেন্ট কিনা সেটা কিভাবে বুঝবে? এখানে ৩ধরণের ঘটনা ঘটতে পারে:

[](http://www.shafaetsplanet.com/planetcoding/wp-content/uploads/2013/06/seg6.png)

কেস A: (b >= i && e <= j) এরকম হলে কারেন্ট সেগমেন্টটা পুরোটাই i-j এর ভিতরে আছে, সেগমেন্টটা রিলেভেন্ট।  
কেস B: (i > e || j < b) এরকম হলে কারেন্ট সেগমেন্টটা পুরোটাই i-j এর বাইরে আছে, এই সেগমেন্টটা নেয়ার দরকার নাই।  
কেস C: কেস A,B সত্য না হলে এই সেগমেন্টের কিছু অংশ i-j এর মধ্যে, সেগমেন্টটাকে আরো ভেঙে নিচে গিয়ে রিলেভেন্ট অংশটা নিতে হবে।

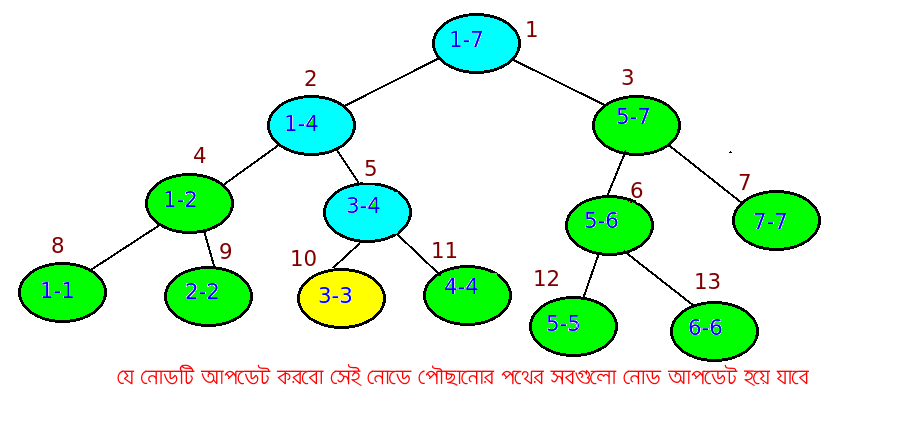
তাহলে আমরা কুয়েরি ফাংশনে প্রতি নোডে গিয়ে দেখবো সেগমেন্টটা রিলেভেন্ট নাকি। যদি রিলেভেন্ট হয় তাহলে সেই নোডের যোগফল রিটার্ণ করবো, যদি বাইরে চলে যায় তাহলে ০ রিটার্ণ করে দিবো, অন্যথায় আমরা সেগমেন্টটা আরো ভেঙে রিলেভেন্ট অংশ খুজবো।



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | int query(int node,int b,int e,int i,int j)  {      if (i > e || j < b)    return 0; //বাইরে চলে গিয়েছে      if (b >= i && e <= j) return tree[node]; //রিলেভেন্ট সেগমেন্ট      int Left=node\*2;  //আরো ভাঙতে হবে      int Right=node\*2+1;      int mid=(b+e)/2;      int p1=query(Left,b,mid,i,j);      int p2=query(Right,mid+1,e,i,j);      return p1+p2; //বাম এবং ডান পাশের যোগফল  } |

init ফাংশনের মতোই কাজ করে কুয়েরি ফাংশনটা। i,j হলো যে রেঞ্জের যোগফল বের করতে হবে সেটা আর b,e হলো কারেন্ট নোডে যে রেঞ্জের যোগফল আছে সেটা।

সবশেষে আপডেট করা, যার জন্য কিউমুলিটিভ সাম ব্যবহার না করে ট্রি বানিয়েছি। তোমাকে বললো i=৩ নম্বর ইনডেক্সের ভ্যালু x=১০ করে দিতে। তারমানে ট্রি তে যেই নোডে ৩-৩ রেঞ্জের যোগফল আছে সেটা আপডেট করে দিবো(নিচের ছবির হলুদ নোড)। নোডটির ভ্যালু আপডেট হলে পথে যেসব নোড ছিলো(নীল নোড) সবগুলোর যোগফল বদলে যাবে, বাকি নোডগুলোর কোনো পরিবর্তন হবেনা কারণ ৩ নম্বর নোড সেগুলো রেঞ্জের বাইরে।

[](http://www.shafaetsplanet.com/planetcoding/wp-content/uploads/2013/06/seg7.png)

আপডেটের কোডেও খুব বেশি পার্থক্য নেই:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | void update(int node,int b,int e,int i,int newvalue)  {      if (i > e || i < b)    return; //বাইরে চলে গিয়েছে      if (b >= i && e <= i) { //রিলেভেন্ট সেগমেন্ট          tree[node]=newvalue; //নতুন মান বসিয়ে দিলাম          return;      }      int Left=node\*2;  //আরো ভাঙতে হবে      int Right=node\*2+1;      int mid=(b+e)/2;      update(Left,b,mid,i,newvalue);      update(Right,mid+1,e,i,newvalue);      tree[node]=tree[Left]+tree[Right];  } |

i নম্বর ইনডেক্সে আপডেট করবো, এক্সট্রা সেগমেন্টগুলো শুরুতেই বাদ দিয়ে দিয়েছি। রিলেভেন্ট সেগমেন্টে গেলে নতুন মান বসিয়ে দিয়েছি, এইখানে কন্ডিশনটা if(b==e) লিখলেও চলতো কারণ সবসময় লিফ নোডে আপডেট করছি আমরা।

সেগমেন্ট ট্রি তো মোটামুটি এই ৩টা ফাংশন সবসময় থাকে init,query,update। অনেক সময় init এর কাজটা আপডেট দিয়ে করে ফেলা যায়। যেমন এখানে তুমি init কল করে ট্রি না বানিয়ে প্রতিটা নোড আলাদা করে আপডেট করে ট্রি বানাতে পারতে। ৩টা ফাংশন মিলিয়ে কোডটা হবে:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57 | #define mx 100001  int arr[mx];  int tree[mx\*3];  void init(int node,int b,int e)  {      if(b==e)      {          tree[node]=arr[b];          return;      }      int Left=node\*2;      int Right=node\*2+1;      int mid=(b+e)/2;      init(Left,b,mid);      init(Right,mid+1,e);      tree[node]=tree[Left]+tree[Right];  }  int query(int node,int b,int e,int i,int j)  {      if (i > e || j < b)    return 0; //বাইরে চলে গিয়েছে      if (b >= i && e <= j) return tree[node]; //রিলেভেন্ট সেগমেন্ট      int Left=node\*2;  //আরো ভাঙতে হবে      int Right=node\*2+1;      int mid=(b+e)/2;      int p1=query(Left,b,mid,i,j);      int p2=query(Right,mid+1,e,i,j);      return p1+p2; //বাম এবং ডান পাশের যোগফল  }  void update(int node,int b,int e,int i,int newvalue)  {      if (i > e || i < b)    return; //বাইরে চলে গিয়েছে      if (b >= i && e <= i) { //রিলেভেন্ট সেগমেন্ট          tree[node]=newvalue;          return;      }      int Left=node\*2;  //আরো ভাঙতে হবে      int Right=node\*2+1;      int mid=(b+e)/2;      update(Left,b,mid,i,newvalue);      update(Right,mid+1,e,i,newvalue);      tree[node]=tree[Left]+tree[Right];  }  int main()  {      READ("in");      int n;      cin>>n;      repl(i,n)          cin>>arr[i];      init(1,1,n);        update(1,1,n,2,0);      cout<<query(1,1,n,1,3)<<endl;      update(1,1,n,2,2);      cout<<query(1,1,n,2,2)<<endl;      return 0;  } |

সেগমেন্ট ট্রি অ্যারেকে বারবার ২ভাগে ভাগ করে, ট্রি এর গভীরতা হবে সর্বোচ্চ log(n) তাই প্রতিটা কুয়েরি আর আপডেটের কমপ্লেক্সিটি O(logn)। init ফাংশনে ট্রি এর প্রতিটা নোডেই একবার যেতে হয়েছে তাই সেক্ষেত্রে কমপ্লেক্সিটি হবে প্রায় O(nlogn)।

সেগমেন্ট ট্রি তুমি তখনই ব্যবহার করতে পারবে যখন দুইটা ছোটো সেগমেন্টকে একসাথে করে বড় সেগমেন্টের ফলাফল বের করা যায়। যোগফল ছাড়াও একটা রেঞ্জের মধ্যে সর্বোচ্চ বা সর্বনিম্ন মান তুমি বের করতে পারবে, বামপাশের সর্বোচ্চ মান এবং ডানপাশের সর্বোচ্চ মান জানলে রুট নোডেরটাও বের করা যায় খুবই সহজে।

এখানে একটা গুরুত্বপূর্ণ জিনিস বাদ পড়েছে। ধরো তোমাকে একটা ইনডেক্সে আপডেট করতে না বলে i থেকে j ইনডেক্সে আপডেট করতে বললো, তাহলে কি করবে? প্রতিটা লিফ নোডে আলাদা করে আপডেট করলে O(nlogn) হয়ে যাবে কমপ্লেক্সিটি যেটা TLE দিবে। এটার জন্য খুবই এলিগেন্ট একটা টেকনিট আছে যার নামে অলস(লেজি) প্রপাগেশন! সে সম্পর্কে পরের পর্বে জানবো, এখন তুমি যতটুকু শিখেছো সেটা দিয়ে [array queries](http://www.lightoj.com/volume_showproblem.php?problem=1082) আর [Curious Robin Hood](http://www.lightoj.com/volume_showproblem.php?problem=1112)প্রবলেমটা সলভ করো।

[পরের পর্ব- লেজি প্রপাগেশন](http://www.shafaetsplanet.com/planetcoding/?p=1591)

১৭টি মন্তব্য

Faiem

১৪ জুনe ২০১৩

খুব ভালো লিখেছেন।

Martuza

১৮ জুনe ২০১৩

This is Straightforward

শাফায়েত

৫ জুলাই ২০১৩

ধন্যবাদ, অ্যাড করে দিলাম লেখাতে।

Sayeed

৫ জুলাই ২০১৩

ট্রি অ্যারের সাইজ হবে ইনপুট অ্যারের ৩গুণ(কেন??)।

Sayeed

৫ জুলাই ২০১৩

query function seems to me inefficient…..in array queries প্রবলেম it will give tle…..a little modification gives faster solution ………

int query(int node,int b,int e,int i,int j)

{

if (i > e || j = i && e <= j) return tree[node]; //????????? ????????

int Left=node\*2; //???? ????? ???

int Right=node\*2+1;

int mid=(b+e)/2;

if(j mid)

{

return query(right,mid+1,end,i,j);

}

else return query(Left,b,mid,i,mid)+query(Right,mid+1,e,mid+1,j); //??? ??? ??? ????? ??????

}

শাফায়েত

৫ জুলাই ২০১৩

না টিএলই দিবেনা, logn এ কাজ করবে এটা, এক্সট্রা সব সেগমেন্ট ফেলে দিবে।

Sayeed

৬ জুলাই ২০১৩

ট্রি অ্যারের সাইজ হবে ইনপুট অ্যারের ৩গুণ………….কেন….???

Sumit

২৮ জুলাই ২০১৩

I think that tree size is double of input array size and double is enough.But why triple????????

শাফায়েত

২৮ জুলাই ২০১৩

ট্রি তে নোড থাকবে দ্বিগুণ সেটা ঠিক আছে কিন্তু অ্যারে সাইজ আরো বেশি নিতে হবে, দ্বিগুণ যথেষ্ট না। ১০০০০০ নোডের একটা সেগমেন্ট ট্রি বানিয়ে init ফাংশন কল করে দেখো “node” ভ্যারিয়েবলটার মান সর্বোচ্চ কত হয়, দেখবে আড়াই গুণের একটু বেশি হয়েছে। এমনকি ৩ গুণও যথেষ্ট না যদি ৩০০০০০ টা নোড থাকে। এখানে একটা অ্যানালাইসিস আছে: http://wcipeg.com/wiki/Segment\_tree#Analysis।

Rajkin Hossain

২৮ আগস্ট ২০১৩

vaiya i am a java coder but i got memory limit exceeded i dont know what to do?? Please help me.. :(

import java.io.\*;

import java.util.\*;

class SegmentTree{

public int [] tree;

public int [] array;

public int sum = 0;

public SegmentTree(int [] y){

tree = new int[y.length\*3];

array = y;

}

public void init(int node,int start,int end)

{

if(start==end)

{

tree[node]= array[start];

return;

}

else{

int Left=node\*2;

int Right=Left+1;

int mid=(start+end)/2;

init(Left,start,mid);

init(Right,mid+1,end);

tree[node]=Math.min(tree[Left],tree[Right]);

}

}

public int query(int node,int start,int end,int i,int j)

{

if(start end || j = i && end 0){

k.readLine();

StringTokenizer s = new StringTokenizer(k.readLine());

int N = Integer.valueOf(s.nextToken());

int [] y = new int[N];

int q = Integer.valueOf(s.nextToken());

StringTokenizer ss = new StringTokenizer(k.readLine());

int c = 0;

while(ss.hasMoreTokens()){

y

1

= Integer.valueOf(ss.nextToken());

}

SegmentTree a = new SegmentTree(y);

a.init(1,0,y.length-1);

z.println("Case "+(test++));

while(q-->0){

StringTokenizer sss = new StringTokenizer(k.readLine());

int i = Integer.valueOf(sss.nextToken());

int j = Integer.valueOf(sss.nextToken());

i--;

j--;

z.println(a.query(1,0,y.length-1, i, j));

}

}

z.flush();

}

}

Rajkin Hossain

২৮ আগস্ট ২০১৩

Sorry to mention thsi problem is from Light oz 1082(Array Queries)

Tahmid Hamim

৯ জানুয়ারি ২০১৪

মেমরী খুব সম্ভবত 2 \* 2^([logN] + 1) লাগে ৩ গুন না।

Bishwajit Purkaystha

৩ মে ২০১৪

“অর্থাৎ রুট x হলে বামেরটা হবে 2x এবং ডানেরটা 2x\*1″

এখানে ডানেরটা 2\*x+1 হতে পারে ।

Shuvo

১১ জুলাই ২০১৪

“ARRAY QUIERIES” can be nicely solved just by using SHAFAET BHAI’s concept described here :)

[ডাটা স্ট্রাকচার: সেগমেন্ট ট্রি-২ (লেজি প্রপাগেশন)](http://www.shafaetsplanet.com/planetcoding/?p=1591)

লেখক: শাফায়েত তারিখ: জুলাই ১৮, ২০১৩ বিভাগ: [প্রোগ্রামিং](http://www.shafaetsplanet.com/planetcoding/?cat=1)

ট্যাগ: [ডাটা স্ট্রাকচার](http://www.shafaetsplanet.com/planetcoding/?tag=%e0%a6%a1%e0%a6%be%e0%a6%9f%e0%a6%be-%e0%a6%b8%e0%a7%8d%e0%a6%9f%e0%a7%8d%e0%a6%b0%e0%a6%be%e0%a6%95%e0%a6%9a%e0%a6%be%e0%a6%b0)• [ডিভাইড এন্ড কনকোয়ার](http://www.shafaetsplanet.com/planetcoding/?tag=%e0%a6%a1%e0%a6%bf%e0%a6%ad%e0%a6%be%e0%a6%87%e0%a6%a1-%e0%a6%8f%e0%a6%a8%e0%a7%8d%e0%a6%a1-%e0%a6%95%e0%a6%a8%e0%a6%95%e0%a7%87%e0%a6%be%e0%a7%9f%e0%a6%be%e0%a6%b0)• [সেগমেন্ট ট্রি](http://www.shafaetsplanet.com/planetcoding/?tag=%e0%a6%b8%e0%a7%87%e0%a6%97%e0%a6%ae%e0%a7%87%e0%a6%a8%e0%a7%8d%e0%a6%9f-%e0%a6%9f%e0%a7%8d%e0%a6%b0%e0%a6%bf)

সেগমেন্ট ট্রির সবথেকে এলিগেন্ট অংশ হলো লেজি প্রপাগেশন টেকনিক। আমরা [আগের পর্বে](http://www.shafaetsplanet.com/planetcoding/www.shafaetsplanet.com/planetcoding/?p=1557) যে সেগমেন্ট ট্রি যেভাবে আপডেট করেছি তাতে একটা বড় সমস্যা ছিলো। আমরা একটা নির্দিষ্ট ইনডেক্স আপডেট করতে পেরেছি, কিন্তু একটা রেঞ্জের মধ্যে সবগুলো ইনডেক্স আপডেট করতে গেলেই বিপদে পরে যাবো। সে কারণেই আমাদের লেজি প্রপাগেশন শিখতে হবে, প্রায় সব সেগমেন্ট ট্রি প্রবলেমেই এই টেকনিকটা কাজে লাগবে। এই পর্বটা পড়ার আগে অবশ্যই তোমাকে সেগমেন্ট ট্রির একদুটি প্রবলেম সলভ করে আসতে হবে, এছাড়া তোমার বুঝতে সমস্যা হবে। [আগের পর্বে](http://www.shafaetsplanet.com/planetcoding/www.shafaetsplanet.com/planetcoding/?p=1557) লেজি প্রপাগেশন দরকার হয়না এমন কয়েকটি প্রবলেম দিয়েছি, আগে সেগুলো সলভ করতে হবে।

তোমাকে একটি অ্যারে দেয়া আছে n সাইজের এবং তোমাকে কুয়েরি করা হচ্ছে **i থেকে j ইনডেক্সের মধ্যে সবগুলো এলিমেন্টের যোগফল বলতে হবে**। আর আপডেট অপারেশনে তোমাকে বলা হলো**i থেকে j ইনডেক্সের মধ্যে সবগুলো সংখ্যার সাথে একটি নির্দিষ্ট সংখ্যা x যোগ করতে**।

যেমন যদি অ্যারেটা শুরুতে হয়তো ছিলো এরকম:

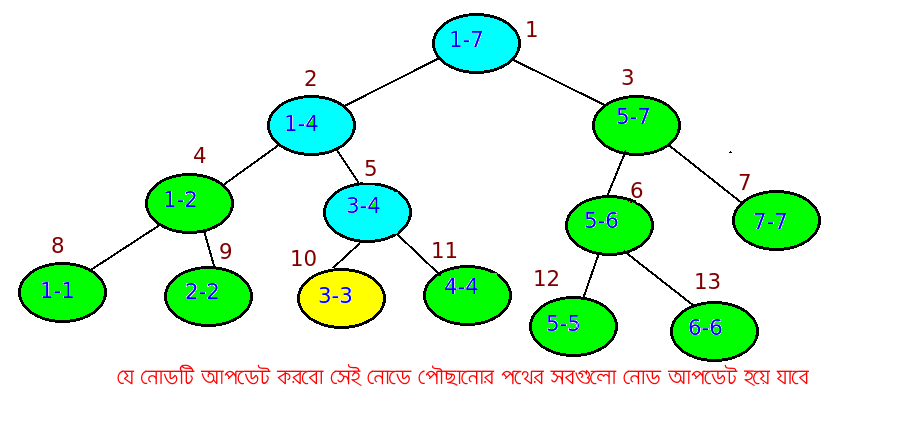
4 1 2 3 9 8 7

তাহলে i=3,j=5 ইনডেক্সের মধ্যে সবগুলো সংখ্যার সাথে ২ যোগ করলে অ্যারেটা হবে:

4 1 2+2 3+2 9+2 8 7

আবার i=3,j=4 ইনডেক্সের মধ্যে সবগুলো সংখ্যার যোগফল হবে ২+২+৩+২=৯।

আগের প্রবলেমে আমরা খালি একটা ইনডেক্স আপডেট করেছিলাম। তখন আমি শুধু ৩ নম্বর ইনডেক্সে আপডেট দেখানোর জন্য এই ছবিটা একেছিলাম:



আমরা সেগমেন্ট ট্রি এর একদম নিচে গিয়ে ৩ যেখানে আছে সেই নোডটা আপডেট করেছি এবং সেই পথের বাকিনোডগুলোকেও আপডেট করে দিয়েছি উপরে ওঠার সময়।

এখন তোমাকে **i=১ এবং j=৪** এই রেঞ্জের সবার সাথে **x** যোগ করতে বলা হলো। একটা সলিউশন হতে পারে তুমি আগের মতো করেই ১,২,৩,৪ ইত্যাদি ইনডেক্স আলাদা করে আপডেট করলে। তাহলে প্রতি আপডেটে কমপ্লেক্সিটি logn এবং সর্বোচ্চ n টি আপডেটের জন্য nlogn সময় লাগবে। এটা খুব একটা ভালো সলিউশন না, আমরা এখানে logn এই আপডেট করতে পারি।

**কিছু ডেফিনেশন:**  
যেকোনো ট্রি স্ট্রাকচারে লিফ নোড হলো সবথেকে নিচের নোডগুলো। লিফ ছাড়া বাকি সবনোড হলো ইন্টারনাল নোড।

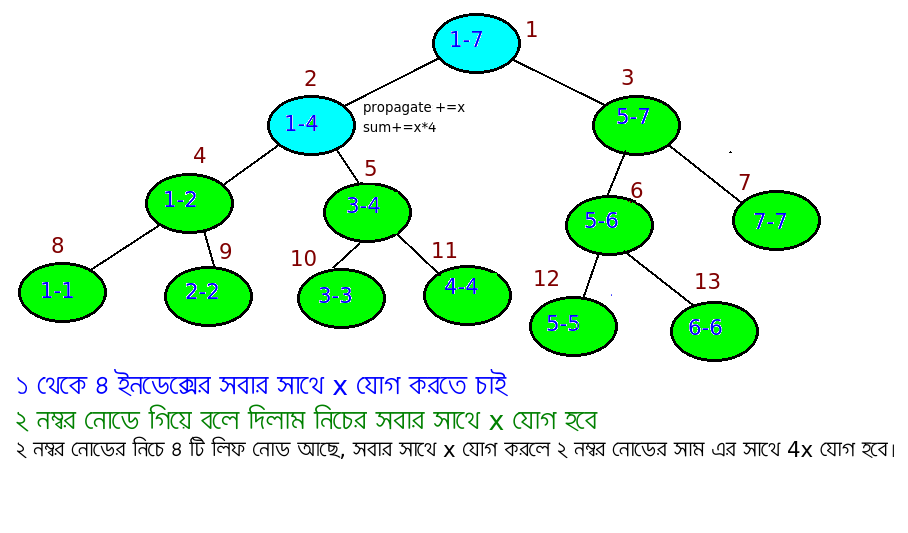
সেগমেন্ট ট্রি তে লিফ নোড গুলোতে কি থাকে? সেখানে থাকে কোনো একটি ইনডেক্সের অরিজিনাল ভ্যালু।

সেগমেন্ট ট্রি তে ইন্টারনাল(লিফ ছাড়া বাকি সব) নোডগুলোতে কি থাকে? সেখানে থাকে নিচে যতগুলো লিফ নোড আছে সবগুলোর মার্জ করা ফলাফল।

সেগমেন্ট ট্রি তে একটা নোডের রেঞ্জ হলো সেই নোডে যেসব ইনডেক্সের মার্জ করা রেজাল্ট আছে। যেমন ছবিতে ৩ নম্বর নোডের রেঞ্জ ৫ থেকে ৭।

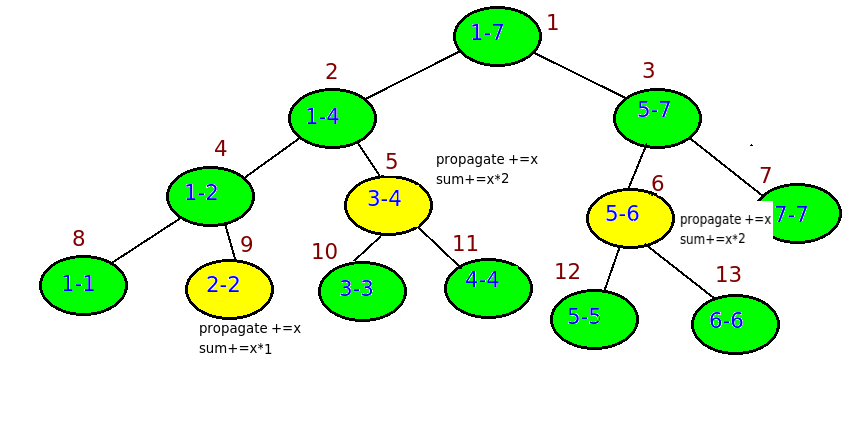
যেমন উপরের ছবিতে ১০ নম্বর নোডে আছে ৩ নম্বর ইনডেক্সের ভ্যালু আর ২ নম্বর নোডে আছে ১,২,৩,৪ নম্বর নোডের যোগফল। কি দরকার এতগুলো লিফ নোডে কষ্ট করে গিয়ে আপডেট করে আসা? তার থেকে আমরা কি ২ নম্বর নোডে এসে সেখানে এই ইনফরমেশনটা রেখে দিতে পারিনা “কারেন্ট নোডের নিচের সবগুলো ইনডেক্সের সাথে x যোগ হবে” ? তার মানে যখন দেখছি **একটা নোডের রেঞ্জ যখন পুরোপুরি কুয়েরির ভিতরে থাকে তখন সেই নোডের নিচে আর না গিয়ে সেখানে প্রপাগেশন ভ্যালুটা সেভ করে রাখতে পারি। একটা নোডের প্রপাগেশন ভ্যালু হলো সেই নোডের রেঞ্জের মধ্যে সব ইনডেক্সের মধ্যে যে ভ্যালুটা যোগ হবে সেটা।**

নিচের ছবিতে দেখো কিভাবে এই এক্সট্রা ইনফরমেশনটা সেভ করা হচ্ছে:

[](http://www.shafaetsplanet.com/planetcoding/wp-content/uploads/2013/07/seg42.png)

প্রতিটি নোডে যোগফল ছাড়াও আরেকটা ভ্যারিয়েবল রাখতে হবে যেটার নাম দিয়েছি propagate। এই ভ্যারিয়েবলটার কাজ হলো তার নিচের লিফ নোডগুলোর সাথে কত যোগ করতে হবে তার হিসাব রাখা। propagate এর মান শুরুতে থাকে শূন্য। এরপর যে রেঞ্জটা আপডেট করতে বলবে সেই রেঞ্জের “রিলেভেন্ট নোড” গুলোতে গিয়ে propagate এর সাথে x যোগ করে আসবো। (আগের পর্বেই জেনেছো “রিলেভেন্ট নোড” হলো যেসব নোডের রেঞ্জ পরোপুরি কুয়েরির ভিতরে আছে)

আরেকটা উদাহরণ, ২ থেকে ৬ নম্বর নোড যদি আপডেট করতে হয় তাহলে হলুদ নোডগুলোতে গিয়ে বলে দিবো নিচের নোডগুলোর সাথে x যোগ করতে:

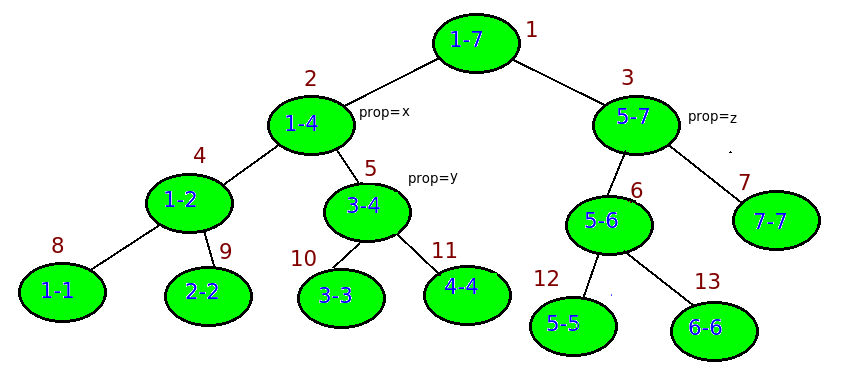
[](http://www.shafaetsplanet.com/planetcoding/wp-content/uploads/2013/07/seg5.png)

আগের আপডেট ফাংশনের মতোই কোনো একটা নোড আপডেট করার পর সেই পথের সবগুলো নোড আপডেট করে উঠতে হবে। আমরা কোডটা দেখি:



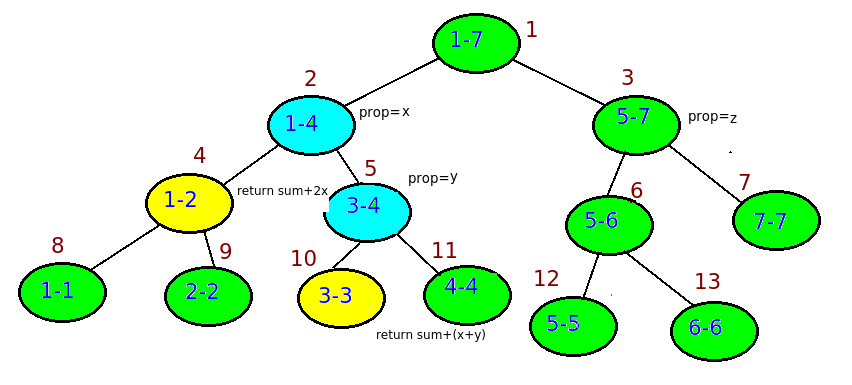
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23 | struct info  {      i64 prop,sum;  }tree[mx\*3]; //sum ছাড়াও নিচে অতিরিক্ত কত যোগ হচ্ছে সেটা রাখবো prop এ  void update(int node,int b,int e,int i,int j,i64 x)  {      if (i > e || j < b)    return;      if (b >= i && e <= j) //নোডের রেঞ্জ আপডেটের রেঞ্জের ভিতরে      {          tree[node].sum+=((e-b+1)\*x); //নিচে নোড আছে e-b+1 টি, তাই e-b+1 বার x যোগ হবে এই রেঞ্জে          tree[node].prop+=x;          //নিচের নোডগুলোর সাথে x যোগ হবে          return;      }      int Left=node\*2;      int Right=(node\*2)+1;      int mid=(b+e)/2;      update(Left,b,mid,i,j,x);      update(Right,mid+1,e,i,j,x);      tree[node].sum=tree[Left].sum+tree[Right].sum+(e-b+1)\*tree[node].prop;          //উপরে উঠার সময় পথের নোডগুলো আপডেট হবে          //বাম আর ডান পাশের সাম ছাড়াও যোগ হবে নিচে অতিরিক্ত যোগ হওয়া মান    } |

আগের আপডেট ফাংশনের সাথে পার্থক্য হলো এখন একটা রেঞ্জে আপডেট করছি এবং কোনো নোডের রেঞ্জ আপডেট রেঞ্জের ভিতরে হলে আমরা নিচে না গিয়ে বলে দিচ্ছি যে নিচের ইনডেক্সগুলোতে x যোগ হবে।

এখন মনে করো আমরা বেশ কয়েকবার আপডেট ফাংশন কল করেছি। নোডগুলোর প্রপাগেটেড ভ্যালুগুলা আপডেট হয়ে নিচের মতো হয়েছে:  
[](http://www.shafaetsplanet.com/planetcoding/wp-content/uploads/2013/07/seg43.png)

যেসব নোডের ভ্যালু লিখিনি সেগুলোতে শুণ্য আছে মনে করো। তাহলে উপরের ছবির মানে হলো ১-৪ ইনডেক্সের সাথে x যোগ হবে, ৫-৭ এর সাথে z যোগ হবে ইত্যাদি।  
এবার প্রশ্ন হলো কুয়েরি করবো কিভাবে?

ধরো আমাদের কুয়েরির রেঞ্জ হলো ১-৩। সাধারণভাবে আমরা আমাদের রিলেভেন্ট নোড ৪ আর ১০ থেকে ভ্যালু নিয়ে যোগ করে দিতাম। কিন্তু আমরা জানি ২ নম্বর নোডের নিচের সবার সাথে x যোগ হয়েছে, তাই ৪ নম্বর নোডের নিচের সবার সাথেও x যোগ হয়েছে! তাই আমরা ৪ নম্বর রেঞ্জে রাখা ভ্যালুর সাথে যোগ করে দিবো 2\*x, কারণ ৪ নম্বর নোডের রেঞ্জে ২টি ইনডেক্স আছে এবং তাদের সবার সাথে x যোগ হয়েছে। ঠিক একই ভাবে যেহেতু ২ এবং ৫ নম্বর নোডের নিচে আছে ১০, তাই ১০ নম্বর নোডের রেঞ্জ ১টি ইনডেক্স আছে এবং তার সাথে যোগ হবে (x+y)।

[](http://www.shafaetsplanet.com/planetcoding/wp-content/uploads/2013/07/seg45.png)

তারমানে এটা পরিষ্কার যে কুয়েরি করা সময় কোনো নোডে যাবার সময় উপরের প্রপাগেটেড ভ্যালু গুলার যোগফল সাথে করে নিয়ে যেতে হবে। তাহলে কুয়েরির ফাংশনে carry নামের একটা প্যারামিটার যোগ করে দেই যার কাজ হবে ভ্যালুটা বয়ে নিয়ে যাওয়া:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | int query(int node,int b,int e,int i,int j,int carry=0)  {      if (i > e || j < b)        return 0;        if(b>=i and e<=j) return tree[node].sum+carry\*(e-b+1);    //সাম এর সাথে যোগ হবে সেই রেঞ্জের সাথে অতিরিক্ত যত যোগ করতে বলেছে সেটা          int Left=node<<1;      int Right=(node<<1)+1;      int mid=(b+e)>>1;        int p1 = query(Left, b,mid,  i, j,   carry+tree[node].prop); //প্রপাগেট ভ্যালু বয়ে নিয়ে যাচ্ছে carry ভ্যারিয়েবল      int p2 = query(Right, mid+1, e, i, j,carry+tree[node].prop);        return  p1+p2;    } |

আগের কোডের সাথে ডিফারেন্স হচ্ছে carry প্যারামিটারে এবং রিটার্ণভ্যালুতে। শুরুতে হিসাব করে রাখা sum এর সাথে যোগ হচ্ছে অতিরিক্ত যে ভ্যালু যোগ হয়েছে সেটা।

মোটামুটি এই হলো লেজি প্রপাগেশনের কাহিনী। সামারী করলে দাড়ায়:

১. রেঞ্জের আপডেট O(logn) এ করতে চাইলে লেজি প্রপাগেশন ব্যবহার করতেই হবে।  
২. লেজি প্রপাগেশনের কাজ হলো লিফ নোডে আপডেট না করে আগেই কোনো নোড আপডেট রেঞ্জের ভিতরে পড়লে সেখানে বলে দেয়া নিচের ইনডেক্সগুলো কিভাবে আপডেট হবে।  
৩. কুয়েরি করার সময় উপরের নোডগুলোতে সেভ করা প্রপাগেশন ভ্যালুগুলো রিলেভেন্ট নোড ক্যারি করে নিয়ে আসতে হবে এবং সেই অনুযায়ী ভ্যালু রিটার্ণ করতে হবে।

অনেক সময় প্রবলেমে বলতে পারে একটা রেঞ্জের মধ্যে সবগুলো সংখ্যাকে x দিয়ে বদলে দিতে (x যোগ নয়), সেক্ষেত্রে প্রপাগেশন ভ্যালু হিসাবে রাখতে হবে নিচের সবনোডকে কোন ভ্যালু দিয়ে বদলে দিতে হবে সেটা। কুয়েরি করার সময় carry ভ্যালুতেও সামান্য পরিবর্তন আসবে, কিরকম পরিবর্তন আসবে সেটা বের করার দায়িত্ব তোমার উপরে ছেড়ে দিলাম।

সেগমেন্ট ট্রির মূল টিউটোরিয়াল এখানেই শেষ। পরবর্তী কোনো পর্বে ২ডি সেগমেন্ট ট্রি এবং কিছু প্রবলেম নিয়ে আলোচনা করবো। টিউটোরিয়ালের কোনো অংশ বুঝতে সমস্যা হলে ইমেইলে যোগাযোগ করতে পারো বা আরো ভালো হয় মন্তব্য অংশে সেটা জানালে।

প্র্যাকটিস প্রবলেম:  
[HORRIBLE](http://www.spoj.com/problems/HORRIBLE/)  
[Lightoj Segment Tree Section](http://www.lightoj.com/volume_problemcategory.php?user_id=1058&category=Segment%20Tree/Interval%20Tree)

[আগের পর্ব](http://www.shafaetsplanet.com/planetcoding/www.shafaetsplanet.com/planetcoding/?p=1557)

[Print Friendly](http://www.printfriendly.com/print?url=http://www.shafaetsplanet.com/planetcoding/?p=1591)

ফেসবুকে মন্তব্য

2 comments

Powered by [Facebook Comments](http://peadig.com/wordpress-plugins/facebook-comments/)

ট্যাগ: [ডাটা স্ট্রাকচার](http://www.shafaetsplanet.com/planetcoding/?tag=%e0%a6%a1%e0%a6%be%e0%a6%9f%e0%a6%be-%e0%a6%b8%e0%a7%8d%e0%a6%9f%e0%a7%8d%e0%a6%b0%e0%a6%be%e0%a6%95%e0%a6%9a%e0%a6%be%e0%a6%b0)• [ডিভাইড এন্ড কনকোয়ার](http://www.shafaetsplanet.com/planetcoding/?tag=%e0%a6%a1%e0%a6%bf%e0%a6%ad%e0%a6%be%e0%a6%87%e0%a6%a1-%e0%a6%8f%e0%a6%a8%e0%a7%8d%e0%a6%a1-%e0%a6%95%e0%a6%a8%e0%a6%95%e0%a7%87%e0%a6%be%e0%a7%9f%e0%a6%be%e0%a6%b0)• [সেগমেন্ট ট্রি](http://www.shafaetsplanet.com/planetcoding/?tag=%e0%a6%b8%e0%a7%87%e0%a6%97%e0%a6%ae%e0%a7%87%e0%a6%a8%e0%a7%8d%e0%a6%9f-%e0%a6%9f%e0%a7%8d%e0%a6%b0%e0%a6%bf)

Sumit

২৮ জুলাই ২০১৩

In this line “tree[node].sum=tree[Left].sum+tree[Right].sum+(e-b+1)\*tree[node].prop;” why I need to add “(e-b+1)\*tree[node].prop;”;I found no change if I write “tree[node].sum=tree[Left].sum+tree[Right].sum;”.

শাফায়েত

২৮ জুলাই ২০১৩

আমার কোডে যেভাবে আপডেট করছি তাতে (e-b+1)\*tree[node].prop; অবশ্যই লাগবে। “A” নোডের নিচের আপডেট করার সময় এটা কনসিডার করা হচ্ছেনা যে A তে জমা থাকা কত এক্সট্রা ভ্যালু নিচে যোগ হচ্ছে(সেটা আমরা কুয়েরিতে কনসিডার করছি)। তাই নিচের নোডের সাম এর সাথে এক্সট্রা ভ্যালুটাও যোগ করে দিচ্ছি। তবে কুয়েরির মতো করে আপডেট ফাংশনেও ভ্যালু ক্যারি করে নিয়ে যাওয়া সম্ভব, সেক্ষেত্রে এইটুকু বাদ দেয়া যাবে তবে অন্য কিছু অংশে পরিবর্তন করতে হবে।

মন্তব্য করুন(বাংলা ভাষায় বাংলা ফন্টে বা ইংরেজি ভাষায় ইংরেজি ফন্টে)

নাম(আবশ্যক)

ইমেইল(আবশ্যক)

ওয়েবসাইট

5 + = 10

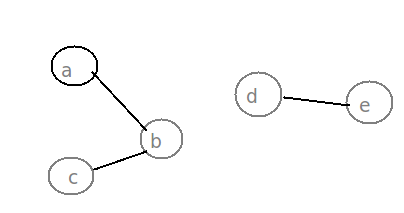
(ইংরেজী ফন্টে বাংলা মন্তব্য মুছে ফেলা হতে পারে)

: [ইউনিয়ন ফাইন্ড](http://www.shafaetsplanet.com/planetcoding/?tag=%e0%a6%87%e0%a6%89%e0%a6%a8%e0%a6%bf%e0%a7%9f%e0%a6%a8-%e0%a6%ab%e0%a6%be%e0%a6%87%e0%a6%a8%e0%a7%8d%e0%a6%a1)• [ট্রি](http://www.shafaetsplanet.com/planetcoding/?tag=%e0%a6%9f%e0%a7%8d%e0%a6%b0%e0%a6%bf)• [ডাটা স্ট্রাকচার](http://www.shafaetsplanet.com/planetcoding/?tag=%e0%a6%a1%e0%a6%be%e0%a6%9f%e0%a6%be-%e0%a6%b8%e0%a7%8d%e0%a6%9f%e0%a7%8d%e0%a6%b0%e0%a6%be%e0%a6%95%e0%a6%9a%e0%a6%be%e0%a6%b0)• [ডিসজয়েন্ট সেট](http://www.shafaetsplanet.com/planetcoding/?tag=%e0%a6%a1%e0%a6%bf%e0%a6%b8%e0%a6%9c%e0%a7%9f%e0%a7%87%e0%a6%a8%e0%a7%8d%e0%a6%9f-%e0%a6%b8%e0%a7%87%e0%a6%9f)

ডাটা স্ট্রাকচার কম্পিউটার সাইন্সের চমতকার অংশগুলোর একটি। আমরা অসংখ্য উপায়ে কম্পিউটারে ডাটা জমা রাখতে পারি। আমরা বাইনারি ট্রি বানাতে পারি,পরে সে গাছ বেয়ে বেয়ে logN এ ডাটা বের করে আনতে পারি,বাসের লাইনের মত কিউ বানাতে পারি,প্রিফিক্স ট্রি বা trie বানিয়ে খুব দ্রুত স্ট্রিং সার্চ করতে পারি। আজ আমরা দেখবো অসাধারণ একটি ডাটা স্ট্রাকচার যার নাম “ডিসজয়েন্ট সেট”।

[kruskal’s MST](http://www.shafaetsplanet.com/planetcoding/?p=825) বা tarjan’s offline LCA ইত্যাদি অ্যালগোরিদম ইমপ্লিমেন্ট করতে ডিসজয়েন্ট সেট খুব গুরুত্বপূর্ণ। এটি ইমপ্লিমেন্ট করতে আমাদের একটি অ্যারে ছাড়া কিছু লাগবেনা।

প্রথমে আমরা দেখবো কি ধরণের কাজে আমাদের এই ডাটা স্ট্রাকচারটি দরকার। মনে করি A,B,C,D,E ৫ জন মানুষ। A-B যদি বন্ধু হয় এবং B-C যদি বন্ধু হয় তাহলে আমরা বলতে পারি A-C ও বন্ধু,অর্থাত তাদের বন্ধুত্ব transitive। এখন আমি বলে দিলাম যে A-B,B-C,D-E এরা পরস্পরের বন্ধু। এখন তোমাকে প্রশ্ন করলাম A-C কি পরস্পরের বন্ধু? B-D কি পরস্পরের বন্ধু? প্রথম প্রশ্নের উত্তর হবে “হ্যা”,২য় প্রশ্নের উত্তর হবে “না”। আমাদের তথ্যগুলোকে গ্রাফের মাধ্যমে দেখাতে পারি:

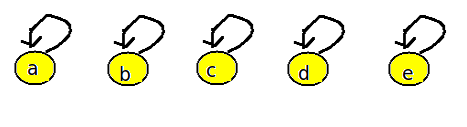


সহজেই বোঝা যাচ্ছে গ্রাফে যেসব নোড একই কম্পোনেন্ট বা সাবগ্রাফের মধ্যে আছে তারা পরস্পরের বন্ধু। তাহলে দু-জন ব্যক্তি বন্ধু নাকি সেটা জানতে হলে আমাদের দেখতে হবে তারা একই সাবগ্রাফে আছে নাকি। সহজেই বিএফএস বা ডিএফএস চালিয়ে এটা বের করা যায়। কিন্তু এগুলোর থেকেও ভালো উপায় হলো “ডিসজয়েন্ট সেট”,কেনো এটা ভালো সেটা কিছুক্ষণ পরেই বুঝতে পারবে।

প্রতিটি মানুষ একটি করে নোড হলে ডিসজয়েন্ট সেট দিয়ে সমাধানের আইডিয়া এরকম:

১. যারা পরস্পরের বন্ধু তারা সবাই একই সেটের অন্তর্গত  
২. যতগুলো সাবগ্রাফ থাকবে ততগুলো সেট থাকবে  
৩. প্রতিটি সেটের একটি representative থাকবে।  
৪. দুটি নোডের representativeএকই হলে তারা একই সেটে আছে। সুতরাং দুজন ব্যক্তি বন্ধু নাকি বুঝতে আমাদের তারা যে সেটে আছে তার representative চেক করতে হবে।

ঘোলাটে লাগছে? সমস্যা নেই,আমরা গ্রাফিকালভাবে এখন ব্যাপারটি বুঝবো,সাথে কোডও দেখবো। শুরুতে কেও কারো বন্ধু নয়। তাহলে সবাই আলাদা আলাদা সেটে আছে। এবং যেহেতু সেটগুলোতে মাত্র একটি করে সদস্য তাই সেই সদস্যটিই হবে সেটের representative। ছবিটি দেখো:



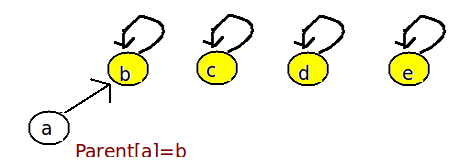
আমরা সেটের representative কে সবসময় **হলুদ রং** দিবো।  
এই অংশটি ইম্প্লিমেন্ট করবো এভাবে:



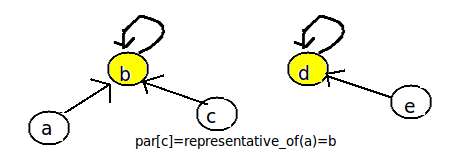
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | //assume that a=1,b=2,c=3,d=4,e=5  int par[10];  void makeset(int n){      par[n]=n; //at first every element is parent of itself.  }  int main()  {      int element=5;      for(int i=1;i<=element;i++)      makeset(i);  } |

par নামক অ্যারেটা দিয়েই আমরা সেট বানাবো। শুরুতে সবার প্যারেন্ট সে নিজেই। প্যারেন্ট বলতে বুঝাচ্ছে নোডটি কার সাথে সংযুক্ত। একটি মাত্র সাধারণ অ্যারে ব্যবহার করে আমরা পুরো স্ট্রাকচারটি বানাবো! এটা একধরনের ট্রি স্ট্রাকচার।

এখন a আর b হঠাত ঠিক করলো যে তারা বন্ধু হবে। তাহলে তাদেরকে একই সেটের মধ্য আসতে হবে। a ঠিক করলো যে b এর representative কে নিজের representative বানিয়ে ফেলবে,তাহলেই তারা একই সেটে চলে আসবে। তাই a এসে b এর সাথে যুক্ত হয়ে গেলো কারণ b এর representative হলো b নিজেই। ছবি দেখো:

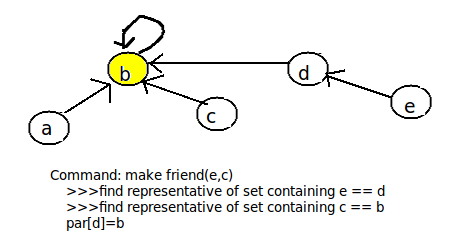


এখন c ভাবলো সেও a এর বন্ধু হবে। এখন c কার নিচে যুক্ত হবে? a এর নিচে c যুক্ত না হবেনা,c যুক্ত হবে a এর representative এর নিচে,তারমানে b এর নিচে।



(d,e কেও এই ফাকে বন্ধু বানিয়ে দিলাম :P )

**পরের ধাপটি ভালো করে দেখো**। এবার আমরা e আর c কে বন্ধু বানাবো। c যেই সেটে সেটার representative হলো b। তার সাথে আমরা যুক্ত করবো e এর representative কে। e এর representative হলো d। b কে d এর প্যারেন্ট বানিয়ে দিলাম। তাহলে b এখন e এরও representative হয়ে গেলো।

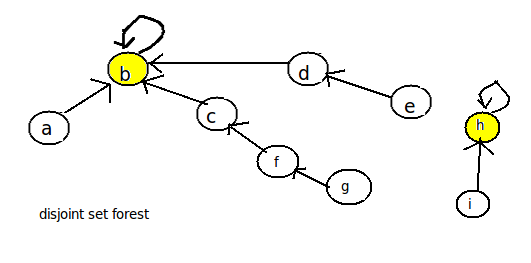


আমরা আবার একটু কোড দেখি:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | void Union(int a,int b)  {      int u=find\_representative(a);      int v=find\_representative(b);      if(u==v) puts("THEY ARE ALREADY FRIENDS");      else      {          par[u]=v; //or you can write par[v]=u too        }  } |

আমরা যে দুটি নোডকে বন্ধু বানাবো তাদের representative বের করলাম find\_representative() ফাংশন কল করে। (ফাংশনটির ডেফিনেশন পরে দেখবো) এবার একটিকে আরেকটির প্যারেন্ট বানিয়ে দিলাম।  
এখন প্রশ্ন হলো representative খুজবো কিভাবে? মনোযোগদিয়ে এই পর্যন্ত পড়ে থাকলে একটা জিনিস চোখে পড়ার কথা,**কোনো একটি সেটের representative element এর প্যারেন্ট সেই এলিমেন্ট নিজেই,অর্থাত এলিমেন্টটি যদি হয় r তাহলে par[r]=r।** আমরা আরেকটু বড় গ্রাফ দেখি। মনে করি a,b,c,d,e এর বন্ধু হতে আরো কয়েকজন চলে এসেছে:



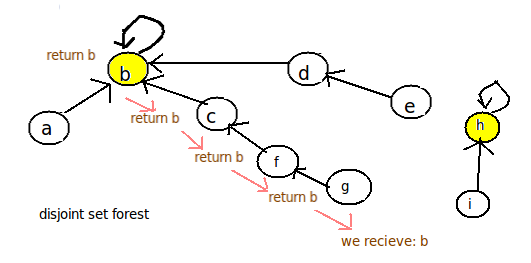
আগের সেটেই শুধু কিছু নোড যুক্ত করা হয়েছে পরের অংশ বোঝার সুবিধার জন্য। ধরি f এর representative বের করতে হবে। f এর প্যারেন্ট সে নিজে নয়,তাহলে তার representative খুজতে তার প্যারেন্টের প্যারেন্ট চেক করি। f এর প্যারেন্ট c। c ও representative নয় কারণ সে নিজেই নিজের প্যারেন্ট নয়,c এর প্যারেন্ট b এবং b নিজেই নিজের প্যারেন্ট। তাহলে b ই হলো representative। ফাংশনটি লিখে ফেলি:



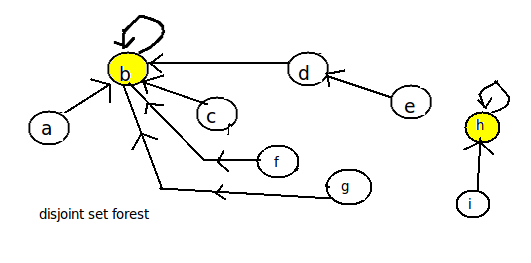
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | int find(int r) // ==find\_representative()  {      if(par[r]==r) return r;  //if r is parent of itself than r is representative      else      {          return find(par[r]); //else find representative of parent of r      }  } |

ফাংশনটি গাছ(tree) বেয়ে উপরে উঠলো যতক্ষননা representative কে খুজে পাওয়া যায়। আগের ফাংশনদুটি(union এবং makeset) তাদের আসল কাজ করেছে O(1) এ। find() ফাংশনে এসে complexity’r প্রশ্ন এসে পড়েছে। worst case এ O(n) টাইম লাগতে পারে উপরের find() ফাংশনের,tree এর depth যত বেশি হবে তত টাইম বেশি লাগবে। ভার্সিটির ল্যাবে লেখা কোডের জন্য এটা ঠিক আছে,কিন্তু কনটেস্টে বা অন্য কোথাও যদি নোড অনেক বেশি হয় তাহলে পারফরম্যান্স খারাপ দিবে।

এখানে একটি চমৎকার optimization আছে। আসলে এই optimization টাই ডিসজয়েন্ট সেট স্ট্রাকচারের সৌন্দর্য। প্রতিটি find() ফাংশন কল এর সাথে সাথে আমরা গ্রাফটি এমন ভাবে পরিবর্তন করবো যেন depth কমে যায়। g এর representative খুজতে আমরা g এর প্যারেন্ট f কে কল দিয়েছি। f তার প্যারেন্টকে কল দিয়ে representative খুজে রিটার্ণ করেছে। রিটার্ণ ভ্যালুগুলোকে এভাবে দেখতে পারি:



প্রতিটি নোডের পাশে রিটার্ণ ভ্যালু লিখে দেয়া হয়েছে। প্রতিটি নোড তার আগের ফাংশন একই ভ্যালু রিটার্ণ করে করে,যেটা সবশেষে আমরা পাই,এই ভ্যালুটাই representative। আমরা return find(par[r]) না লিখে যদি লিখতাম return par[r]=find(par[r]) তাহলে কি ঘটতো? রিটার্ণ করার সময় রিটার্ণ ভ্যালুটাকে নিজের প্যারেন্ট বানিয়ে ফেলতো,অর্থাত সবাই **representative কেই বানিয়ে ফেলতো নিজের প্যারেন্ট!** ফাংশন কলের সাথে সাথে গ্রাফটি হয়ে যেত এমন:



ট্রি এর depth কমে গিয়েছে,তাই নয় কি?এটাকে বলা হয় path compression। পরে আমরা যখন f এর representative খুজবো তখন আর মাত্র একটি ডাল(edge) বেয়ে উঠতে হবে। wiki তে সুন্দর করে লেখা আছে:

path compression, is a way of flattening the structure of the tree whenever Find is used on it. The idea is that each node visited on the way to a root node may as well be attached directly to the root node; they all share the same representative. To effect this, as Find recursively traverses up the tree, it changes each node’s parent reference to point to the root that it found. The resulting tree is much flatter, speeding up future operations not only on these elements but on those referencing them, directly or indirectly.

মোটামুটি এই হলো আমাদের disjoint set। দুটি নোড একই সেটে আছে নাকি চেক করতে আমরা তাদের representative বের করে তারা সমান নাকি সেটা চেক করবো। দুটি নোড কে একই সেটে নিতে হলে তাদের representative বের করে একটিকে আরেকটির প্যারেন্ট বানিয়ে দিবো। [kruskal এ](http://www.shafaetsplanet.com/planetcoding/?p=825) দুটি নোড একই সাব-ট্রিতে আছে নাকি সেটা চেক করতে disjoint set ব্যবহার করা হয়। একই tree তে না থাকলে union ফাংশনটি কল করে এক করা হয়।

এখন যদি সত্যিই কিছু শিখতে চাও তাহলে ঝটপট কিছু প্রবলেম সলভ করে ফেলো:  
[Graph connectivity](http://uva.onlinejudge.org/external/4/459.html)  
[Nature](http://uva.onlinejudge.org/external/106/10685.html)  
[Virtual Friend(\*)](http://uva.onlinejudge.org/external/115/11503.html)

আরো জানতে চাইলে:  
[টপকোডার টিউটোরিয়াল](http://community.topcoder.com/tc?module=Static&d1=tutorials&d2=disjointDataStructure)  
[Wikipedia](http://en.wikipedia.org/wiki/Disjoint-set_data_structure)

[Print Friendly](http://www.printfriendly.com/print?url=http://www.shafaetsplanet.com/planetcoding/?p=763)

ফেসবুকে মন্তব্য

3 comments

Powered by [Facebook Comments](http://peadig.com/wordpress-plugins/facebook-comments/)

ট্যাগ: [ইউনিয়ন ফাইন্ড](http://www.shafaetsplanet.com/planetcoding/?tag=%e0%a6%87%e0%a6%89%e0%a6%a8%e0%a6%bf%e0%a7%9f%e0%a6%a8-%e0%a6%ab%e0%a6%be%e0%a6%87%e0%a6%a8%e0%a7%8d%e0%a6%a1)• [ট্রি](http://www.shafaetsplanet.com/planetcoding/?tag=%e0%a6%9f%e0%a7%8d%e0%a6%b0%e0%a6%bf)• [ডাটা স্ট্রাকচার](http://www.shafaetsplanet.com/planetcoding/?tag=%e0%a6%a1%e0%a6%be%e0%a6%9f%e0%a6%be-%e0%a6%b8%e0%a7%8d%e0%a6%9f%e0%a7%8d%e0%a6%b0%e0%a6%be%e0%a6%95%e0%a6%9a%e0%a6%be%e0%a6%b0)• [ডিসজয়েন্ট সেট](http://www.shafaetsplanet.com/planetcoding/?tag=%e0%a6%a1%e0%a6%bf%e0%a6%b8%e0%a6%9c%e0%a7%9f%e0%a7%87%e0%a6%a8%e0%a7%8d%e0%a6%9f-%e0%a6%b8%e0%a7%87%e0%a6%9f)

: [ট্রাই](http://www.shafaetsplanet.com/planetcoding/?tag=%e0%a6%9f%e0%a7%8d%e0%a6%b0%e0%a6%be%e0%a6%87)• [ট্রি](http://www.shafaetsplanet.com/planetcoding/?tag=%e0%a6%9f%e0%a7%8d%e0%a6%b0%e0%a6%bf)• [ডাটা স্ট্রাকচার](http://www.shafaetsplanet.com/planetcoding/?tag=%e0%a6%a1%e0%a6%be%e0%a6%9f%e0%a6%be-%e0%a6%b8%e0%a7%8d%e0%a6%9f%e0%a7%8d%e0%a6%b0%e0%a6%be%e0%a6%95%e0%a6%9a%e0%a6%be%e0%a6%b0)• [প্রিফিক্স ট্রি](http://www.shafaetsplanet.com/planetcoding/?tag=%e0%a6%aa%e0%a7%8d%e0%a6%b0%e0%a6%bf%e0%a6%ab%e0%a6%bf%e0%a6%95%e0%a7%8d%e0%a6%b8-%e0%a6%9f%e0%a7%8d%e0%a6%b0%e0%a6%bf)

ইন্টারনেট যারা ব্যবহার করে তারা সবাই মনে হয় কখনো না কখনো এটা ভেবে অবাক হয়েছে যে গুগলে সার্চ করলে কিভাবে এত তথ্য চোখের নিমিষে সামনে চলে আসে! চিন্তা করে দেখো হাজার কিলোমিটার দূরে দানবাকৃতির ডাটাবেস থেকে প্রয়োজনীয় ডাটা খুঁজে তোমার কাছে পাঠিয়ে দিচ্ছে কয়েক সেকেন্ডের মধ্যে। শুধুমাত্র শক্তিশালী হার্ডওয়্যার, দ্রুতগতির ইন্টারনেট দিয়ে এটা সম্ভব না, এর পিছে আছে দারুণ কিছু সার্চিং অ্যালগোরিদম এবং ডাটা স্ট্রাকচার।

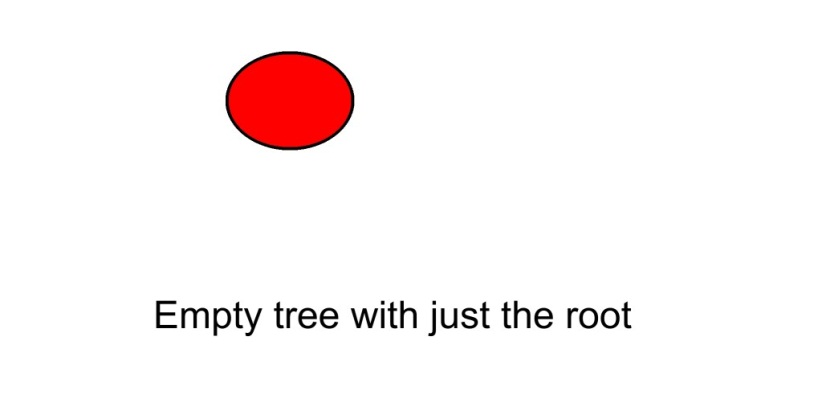
আমরা সেসব জটিল জিনিসে যাবোনা, আমরা আজকে শিখবো খুব সহজে ইমপ্লিমেন্ট করা যায় এমন একটা ডাটা স্ট্রাকচার যেটা ব্যবহার করে তুমি খুব দ্রুত অনেক অনেক নাম এর মধ্য থেকে একটা নাম খুজে বের করতে পারবে, ডাটাবেসে যত মিলিয়ন নামই থাকুক না কেন তুমি যে নামটা খুজে বের করতে চাও সেটাতে যতগুলো অক্ষর আছে সেটার উপর নির্ভর করবে কত সময় লাগবে সেটা খুজে বের করতে। তারমানে এখানে তোমার সার্চিং কমপ্লেক্সিটি ডাটাবেস সাইজের উপর নির্ভরশীল না, দারুণ একটা ব্যাপার তাইনা?

ধরো তোমাকে একটা ডিকশনারী দেয়া হলো যেখানে নিচের শব্দগুলো আছে:

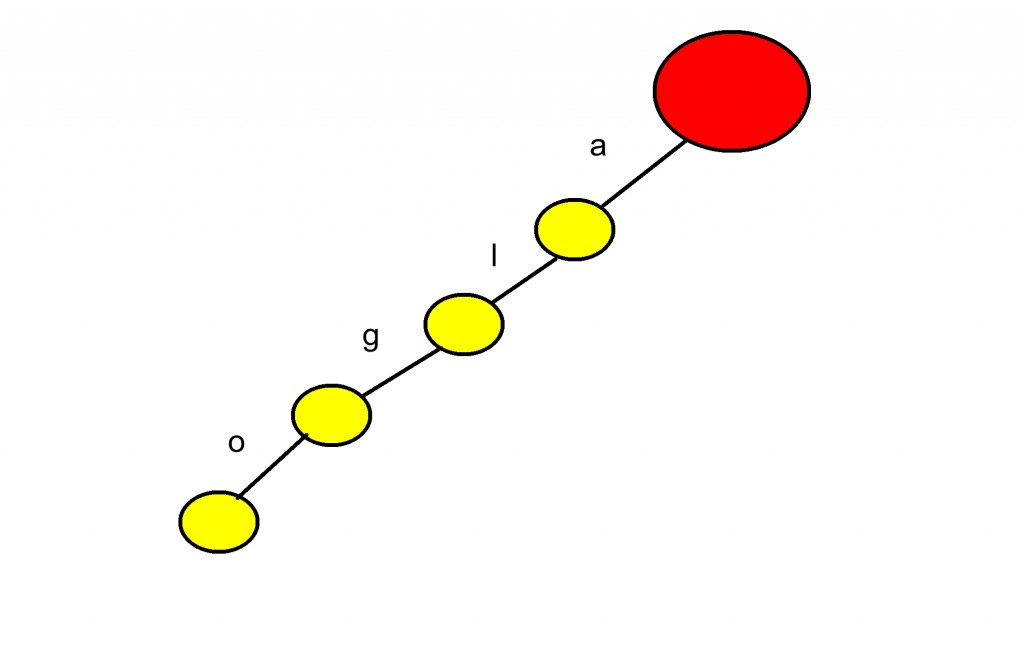
algo  
algea  
also  
tom  
to

এখন আমরা এই ডিকশনারিটাকে এমনভাবে মেমরিতে রাখতে চেষ্টা করবো যেন খুব সহজে কোনো একটা শব্দ খুজে পাওয়া যায়। একটা উপায় হলো শব্দগুলোকে সর্ট করে রাখা যেটা রিয়েল লাইফে কাগজের ডিকশনারি গুলোতে রাখা হয়, তাহলে বাইনারি সার্চ করেই আমরা কোনো একটা শব্দ খুজে বের করতে পারবো। আরেকটা উপায় হলো প্রিফিক্স ট্রি বা সংক্ষেপে ট্রাই(trie) ব্যবহার করা। trie শব্দটা এসেছে “re**trie**val” শব্দটা থেকে। সেই হিসাবে এটার উচ্চারণ “ট্রি” হওয়ার কথা কিন্তু গ্রাফ থিওরীতে ট্রি এর আরো ব্যপক ব্যবহার আছে তাই এটাকে বলা হয় “ট্রাই” । **প্রিফিক্স মানে হলো একটা স্ট্রিং এর শুরু থেকে কয়েকটা ক্যারেকটার নিয়ে নতুন স্ট্রিং তৈরি করা। যেমন blog এর প্রিফিক্স হলো b,bl,blo এবং blog।**

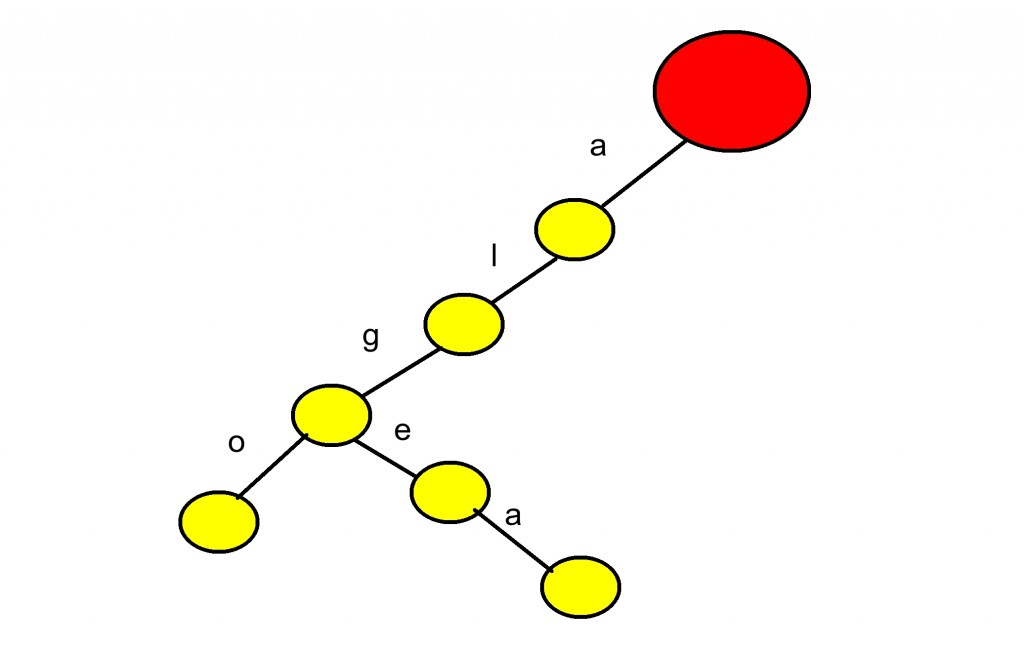
শুরুতে আমাদের একটা রুট নোড ছাড়া কিছুই নেই।

[](http://www.shafaetsplanet.com/planetcoding/wp-content/uploads/2013/12/trie1.jpeg)

এখন আমরা algo শব্দটাকে যোগ করবো। নিচের ছবিতে দেখো কিভাবে শব্দটা যোগ করা হয়েছে। রুট নোড থেকে আমরা একটা এজ দিবো যেই এজ এর নাম হবে “a”। তারপর নতুন তৈরি হওয়া নোড থেকে “l” নামের একটা এজ তৈরি করবো। এভাবে “g” আর “o” এজ দুইটাও তৈরি করবো। লক্ষ্য করো নোডে আমরা কোনো তথ্য রাখছিনা, খালি নোড থেকে এজ বের করছি।

[](http://www.shafaetsplanet.com/planetcoding/wp-content/uploads/2013/12/trie11.jpeg)

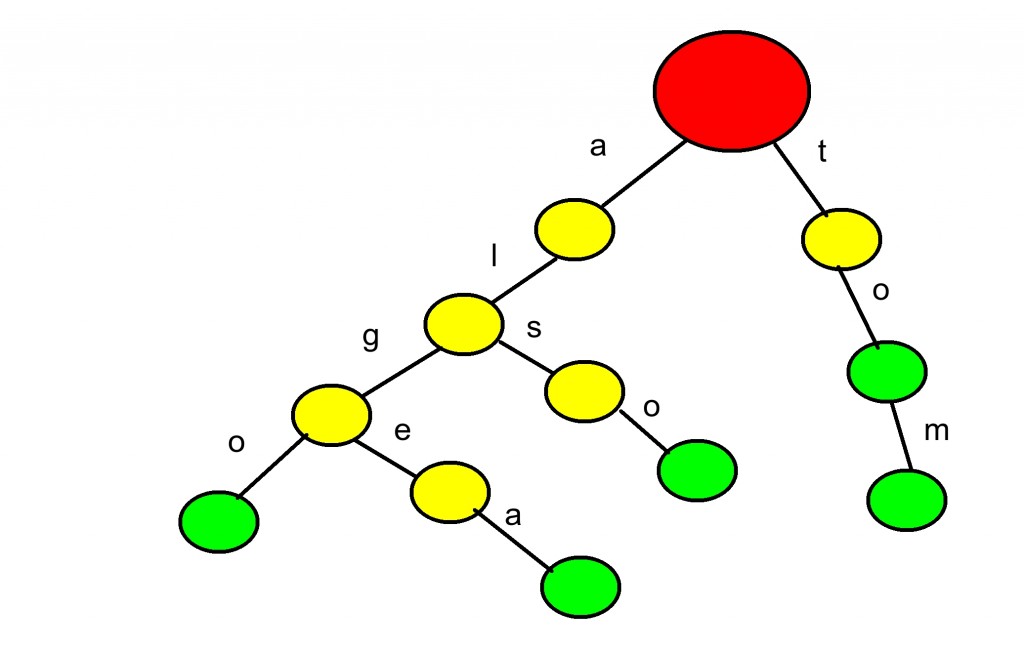
এখন আমরা algea শব্দটা যোগ করতে চাই। রুট থেকে “a” নামের এজ দরকার, সেটা অলরেডি আছে, নতুন করে যোগ করা দরকার নাই। ঠিক সেরকম a থেকে l এবং l থেকে g তেও এজ আছে। তারমানে “alg”  অলরেডি ট্রাই তে আছে, আমরা শুধু e আর a যোগ করবো।

[](http://www.shafaetsplanet.com/planetcoding/wp-content/uploads/2013/12/trie12.jpeg)

also শব্দটাকে যোগ করবো এবার। রুট থেকে “al” প্রিফিক্স এরইমধ্যে আছে, শুধু “so” যোগ করতে হবে।

এবার “tom” যোগ করি। এবার রুট থেকে নতুন এজ তৈরি করতে হবে কারণ tom এর কোনো প্রিফিক্স আগে যোগ করা হয়নি।

এখন “to” শব্দটা যোগ করবো কিভাবে? “to” পুরোপুরি tom এর প্রিফিক্স তাই নতুন কোনো এজ যোগ করা দরকার নাই। আমরা যে কাজটা করতে পারি সেটা বলে নোডগুলোতে কিছু এন্ড-মার্ক বসানো। যেসব নোডে এসে অন্তত একটা শব্দ কমপ্লিট হয়েছে সেসব নোডে আমরা এন্ডমার্ক বসিয়ে দেই, ছবিতে সবুজ রঙ হলো এন্ডমার্ক। আগের সব শব্দের জন্য এবং সেই সাথে নতুন শব্দ “to” এর জন্য এন্ডমার্ক বসালে ট্রাইটা এরকম দেখাবে:

[](http://www.shafaetsplanet.com/planetcoding/wp-content/uploads/2013/12/trie13.jpeg)

নিশ্চয়ই বুঝতে পারছো সবুজ মার্কগুলো কেন বসিয়েছি। মার্ক দেখে সহজেই বোঝা যাচ্ছে কোন কোন শব্দ ট্রাইতে আছে। কোন ক্যারেকটার নিচ্ছি সেই তথ্য থাকবে এজ এ, আর এন্ডমার্কগুলো থাকবে নোড এ।

এভাবে শব্দগুলো রাখার সুবিধা কি? ধরো তোমাকে বলা হলো “alice” শব্দটা ডিকশনারিতে আছে কিনা বলতে। তুমি শুরু থেকে ট্রাই ধরে আগাতে থাকো। প্রথমে দেখো রুট থেকে a নামের এজ আছে নাকি, তারপর চেক করো a থেকে l নামের এজ আছে নাকি। এরপরে l থেকে i নামের এজ খুজে পাওয়া যাচ্ছেনা, তাই বলতে পারো alice শব্দটা নেই।

“alg” শব্দটা খুজতে দিলে তুমি root->a, a->l এবং l->g এজগুলো সবই খুজে পাবে, কিন্তু শেষ পর্যন্ত কোনো সবুজ নোডে যেতে পারবেনা, তারমানে alg ও ডিকশনারিতে নেই। “tom” খুজতে গেলে তুমি একটা সবুজ নোডে গিয়ে শেষ করবে তাই শব্দটা ডিকশনারিতে আছে।

ট্রাই ইমপ্লিমেন্ট করার সহজ একটা উপায় হলো লিংকড লিস্ট ব্যবহার করা। লিংকড লিস্ট, পয়েন্টার এসব শুনে ভয়ের কিছুই নেই, তুমি যদি লিংকলিস্ট ব্যবহার করতে অভ্যস্ত নাও হও আশা করি এই ইমপ্লিমেন্টেশনটা দেখে শিখে ফেলতে পারবে।আমাদের প্রতিটা নোডে ২টি জিনিস থাকবে:

**১.**এন্ড-মার্ক রাখার জন্য একটা ভ্যারিয়েবল।

**২.**প্রতিটা নোড থেকে a,b,c,……..,x,y,z ইত্যাদি নামের এজ তৈরি হতে পারে।  আমরা প্রতিটা ক্যারেক্টারের জন্য একটা পয়েন্টার রাখবো। পয়েন্টারের সাহায্যে একটা নোড আরেকটার সাথে যোগ হবে। a নামের পয়েন্টার দিয়ে যোগ হলে বুঝতে হবে কারেন্ট নোড থেকে a নামের একটা এজ আছে। শুরুতে সবগুলো পয়েন্টার “নাল” থাকবে।

আমরা প্রথমেই নোডটা তৈরি করে ফেলি:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | struct node  {      bool endmark;      node \*next[26+1];      node()      {          endmark=false;          for(int i=0;i<26;i++) next[i]=NULL;      }  }\*root;  int main(){      root=new node();      return 0;  } |

“next[]” অ্যারের প্রতিটা এলিমেন্ট আরেকটা নোডকে পয়েন্ট করে। next[0] দিয়ে নতুন নোডকে পয়েন্ট করা হলে সেই এজ এর নাম “a”, next[1] এর জন্য এজ এর নাম “b”, next[25] এর জন্য “z”। শুরুতে সবগুলো পয়েন্টার নাল।

লক্ষ্য করো নোডের ভিতর একটা কনস্ট্রাক্টর “node()” তৈরি করেছি। যখনই নতুন নোড তৈরির জন্য new node() কল করবো তখনই ভ্যারিয়েবলগুলোকে শূন্য বা নাল বানিয়ে দিবে। এটা না দিলে গার্বেজ ভ্যালু থাকতো। root ভ্যারিয়েবলটা হলো আমাদের রুট নোড, উপরের ছবিগুলোতে লাল রঙ এর নোড। আসলে রুট একটা পয়েন্টার, যখন root=new node(); লাইনটা এক্সিকিউট হবে তখনই একটা নতুন নোড তৈরি করে root যে মেমরি অ্যাড্রেসকে পয়েন্ট করে সেখানে অ্যাসাইন করে দেয়া হবে। এটাকে একটু গালভরা ভাষায় বলে “instance” তৈরি করা।

এবার আমাদের একটা ফাংশন লাগবে নতুন শব্দ ট্রাইতে যোগ করার জন্য:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | void insert(char \*str,int len)  {      node \*curr=root;      for(int i=0;i<len;i++)      {          int id=str[i]-'a';          if(curr->next[id]==NULL)              curr->next[id]=new node();          curr=curr->next[id];      }      curr->endmark=1;    } |

রুট ভ্যারিয়েবলটা আমাদের সবসময় দরকার হবে তাই “curr” এর মধ্যে সেটার কপি তৈরি করে কাজ করি। যেহেতু পয়েন্টার নিয়ে কাজ করছি তাই রুট থেকে নতুন এজ তৈরি করা আর “curr” থেকে নতুন এজ তৈরি করা একই কথা। আমরা এখন শুধু a-z নিয়ে কাজ করছি, তাই অ্যাসকি ভ্যালুগুলোকে 0-25 এ কনভার্ট করে নিবো ‘a’ এর অ্যাসকি ভ্যালু বিয়োগ করে। insert করা খুব সহজ, আমরা শুধু চেক করবো কারেন্ট নোড (curr) থেকে বর্তমানে যে ক্যারেকটারে আছি সেই নামের কোনো এজ আছে নাকি, না থাকলে নতুন নোড তৈরি করতে হবে। এরপরে সেই এজ ধরে আমরা পরের নোডে যাবো। সবার শেষ নোডটায় এন্ড-মার্ক true করে দিবো।

ইনসার্ট করার পর এখন সার্চ করবো। এটা আসলে ঠিক ইনসার্ট এর মতোই। পার্থক্য হলো যে এজটা দরকার সেটা না থাকলে তৈরি করে নিচ্ছিলাম, এখন এজ না থাকলে false রিটার্ণ করে দিবো।



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | bool search(char \*str,int len)  {      node \*curr=root;      for(int i=0;i<len;i++)      {          int id=str[i]-'a';          if(curr->next[id]==NULL) return false;          curr=curr->next[id];      }      return curr->endmark;  } |

লক্ষ্য করো সবকাজ ইনসার্ট এর মতোই করেছি। সবার শেষে লাস্ট নোডটার এন্ডমার্ক রিটার্ণ করে দিয়েছি। এন্ডমার্ক true হলে শব্দটা আছে, false হলে নাই।

আমাদের মূল কোড শেষ। আমরা এখন যেকোনো শব্দ ট্রাইতে যোগ করতে পারবো, আবার ট্রাই থেকে কোনো শব্দ খুজতে পারবো। অনেক সময় প্রতিটা টেস্টকেস এর জন্য ট্রাই তৈরি করতে গেলে মেমরি লিমিট নিয়ে সমস্যা হয়। তাই নিরাপদ উপায় হলো প্রতি কেস এর পর ব্যবহৃত মেমরি-সেল গুলোকে ডিলিট করে দেয়া। শুধু root ডিলিট করলে হবেনা, প্রতিটা নোড করতে হবে। আমরা সে জন্য একটা রিকার্সিভ ফাংশন লিখতে পারি:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | void del(node \*cur)  {        for(int i=0;i<26;i++)               if(cur->next[i])                    del(cur->next[i]) ;            delete(cur) ;  } |

এই ফাংশনটা প্রতিটা নোডে গিয়ে আগে চাইল্ডগুলোকে ডিলিট করে এসে তারপর নোডটাকে ডিলিট করে দিবে।

সম্পূর্ণ কোডটা এরকম:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69 | struct node  {      bool endmark;      node \*next[26+1];      node()      {          endmark=false;          for(int i=0;i<26;i++) next[i]=NULL;      }  }\*root;    void insert(char \*str,int len)  {      node \*curr=root;      for(int i=0;i<len;i++)      {          int id=str[i]-'a';          if(curr->next[id]==NULL)              curr->next[id]=new node();          curr=curr->next[id];      }      curr->endmark=true;    }  bool search(char \*str,int len)  {      node \*curr=root;      for(int i=0;i<len;i++)      {          int id=str[i]-'a';          if(curr->next[id]==NULL) return false;          curr=curr->next[id];      }      return curr->endmark;  }  void del(node \*cur)  {        for(int i=0;i<26;i++)               if(cur->next[i])                    del(cur->next[i]) ;               delete(cur) ;  }  int main(){        puts("ENTER NUMBER OF WORDS");      root=new node();      int num\_word;      cin>>num\_word;      for(int i=1;i<=num\_word;i++)      {          char str[50];          scanf("%s",str);          insert(str,strlen(str));      }      puts("ENTER NUMBER OF QUERY";);      int query;      cin>>query;      for(int i=1;i<=query;i++)      {          char str[50];          scanf("%s",str);          if(search(str,strlen(str)))    puts("FOUND");          else puts("NOT FOUND");      }      del(root); //ট্রাইটা ধ্বংস করে দিলাম      return 0;  } |

**কমপ্লেক্সিটি:** প্রতিটা শব্দ খুজতে লুপ চালাতে হচ্ছে শব্দটার লেংথ পর্যন্ত, সার্চিং এর কমপ্লেক্সিটি O(length)। প্রতিটা শব্দ ইনসার্ট করার কমপ্লেক্সিটিও একই। মেমরি কতখানি লাগবে সেটা ডিপেন্ড করে ইমপ্লিমেন্টেশন এবং শব্দগুলোর প্রিফিক্স কতখানি ম্যাচ করে তার উপর। উপরের ইমপ্লিমেন্টেশন দিয়ে প্রায় 10^6 টা ক্যারেকটার ট্রাইতে ইনসার্ট করা যাবে (10^6 টা ওয়ার্ড নয়, ক্যারেকটার বা লেটার)।

তুমি চাইলে ট্রাই লিংকলিস্ট ছাড়া সাধারণ অ্যারে ব্যবহার করে ইমপ্লিমেন্ট করতে পারো, নিজে চেষ্টা করো!

**ট্রাই এর কিছু ব্যবহার:**

১. একটা ডিকশনারিতে অনেকগুলো শব্দ আছে, কোনো একটা শব্দ আছে নাকি নাই খুজে বের করতে হবে। এই প্রবলেমটা আমরা উপরের কোডেই সলভ করেছি।

২. ধরো তোমার ৩ বন্ধুর টেলিফোন নম্বর হলো “৫৬৭৮”, “৪৩২২”, “৫৬৭”। তুমি যখন প্রথম বন্ধুকে ডায়াল করবে তখন ৫৬৭ চাপার সাথে সাথে ৩য় বন্ধুর কাছে ফোন চলে যাবে কারণ ৩য় বন্ধুর নাম্বার প্রথম জনের প্রিফিক্স! অনেকগুলো ফোন নম্বর দেয়া আছে, বলতে হবে এরকম কোনো নম্বর আছে নাকি যেটা অন্য নম্বরের প্রিফিক্স। [(UVA 11362)](http://uva.onlinejudge.org/external/113/11362.html)।

৩. একটা ডিকশনারিতে অনেকগুলো শব্দ আছে। এখন কোনো একটা শব্দ কয়বার “prefix” হিসাবে এসেছে সেটা বের করতে হবে। যেমন “al” শব্দটা উপরের ডিকশনারিতে ৩বার প্রিফিক্স হিসাবে এসেছে (algo, algea, also এই সবগুলো শব্দের প্রিফিক্স “al”)। এটা বের করার জন্য প্রতিটা নোডে একটা কাউন্টার ভ্যারিয়েবল রাখতে হবে, কোনো নোডে যতবার যাবে ততবার কাউন্টারের মান বাড়িয়ে দিবে। সার্চ করার সময় প্রিফিক্সটা খুজে বের করে কাউন্টারের মান দেখবে।

৪. মোবাইলের ফোনবুকে সার্চ করার সময় তুমি যখন কয়েকটা লেটার লিখো তখন সেই প্রিফিক্স দিয়ে কি কি নাম শুরু হয়েছে সেগুলো সাজেশন বক্সে দেখায়। এটা তুমি ট্রাই দিয়ে ইমপ্লিমেন্ট করতে পারবে?

৪. দু্টি স্ট্রিং এর “longest common substring” বের করতে হবে। (subsequence হলে ডিপি দিয়ে সহজে করা যায়, এখানে substring চেয়েছি)।  
(হিন্টস: একটা স্ট্রিং এর শেষ থেকে এক বা একাধিক ক্যারেকটার নেয়া হলে সেটাকে স্ট্রিংটার সাফিক্স বলে, যেমন blog এর সাফিক্স g,og,log,blog। আর প্রতিটা substring ই কিন্তু কোনো না কোনো সাফিক্স এর প্রিফিক্স!! তাই সবগুলো সাফিক্সকে ট্রাইতে ইনসার্ট করলে কাজটা সহজ হয়ে যায়!)

৫. (অ্যাডভান্সড) সম্ভবত ২০১১তে ডেফোডিল ইউনিভার্সিটির ন্যাশনাল কনটেস্টে এসেছিলো প্রবলেমটা। একটা ডিকশনারি ইনপুট দেয়া থাকবে। প্রতিবার ডিকশনারির ২টা শব্দ কুয়েরি দিবে, বলতে হবে তাদের মধ্যে common prefix এর দৈর্ঘ্য কত। যেমন algo আর algea এর কমন প্রিফিক্স alg, দৈর্ঘ্য ৩। ট্রাইতে ডিকশনারিতে ইনসার্ট করে প্রতি কুয়েরিতে শব্দদুটি এন্ড-মার্ক থেকে LCA(lowest common ancestor) বের করে প্রবলেমটা সলভ করা যায়।

কিছু প্রবলেম:  
[UVA 10226](http://uva.onlinejudge.org/external/102/10226.html)  
[(UVA 11362 Phonebook)](http://uva.onlinejudge.org/external/113/11362.html)  
[UVA 11488 Hyper prefix sets](http://uva.onlinejudge.org/external/114/11488.html)  
[POJ 2001 Shortest Prefix](http://poj.org/problem?id=2001)  
[POJ 1056](http://poj.org/problem?id=1056)

হ্যাপি কোডিং!

[Print Friendly](http://www.printfriendly.com/print?url=http://www.shafaetsplanet.com/planetcoding/?p=1679)

ফেসবুকে মন্তব্য

5 comments

Powered by [Facebook Comments](http://peadig.com/wordpress-plugins/facebook-comments/)

ট্যাগ: [ট্রাই](http://www.shafaetsplanet.com/planetcoding/?tag=%e0%a6%9f%e0%a7%8d%e0%a6%b0%e0%a6%be%e0%a6%87)• [ট্রি](http://www.shafaetsplanet.com/planetcoding/?tag=%e0%a6%9f%e0%a7%8d%e0%a6%b0%e0%a6%bf)• [ডাটা স্ট্রাকচার](http://www.shafaetsplanet.com/planetcoding/?tag=%e0%a6%a1%e0%a6%be%e0%a6%9f%e0%a6%be-%e0%a6%b8%e0%a7%8d%e0%a6%9f%e0%a7%8d%e0%a6%b0%e0%a6%be%e0%a6%95%e0%a6%9a%e0%a6%be%e0%a6%b0)• [প্রিফিক্স ট্রি](http://www.shafaetsplanet.com/planetcoding/?tag=%e0%a6%aa%e0%a7%8d%e0%a6%b0%e0%a6%bf%e0%a6%ab%e0%a6%bf%e0%a6%95%e0%a7%8d%e0%a6%b8-%e0%a6%9f%e0%a7%8d%e0%a6%b0%e0%a6%bf)

সান

২৭ ডিসেম্বর ২০১৩

ভাইয়া , সব শেষের পিকচার এ tom এর o এর কালার মনে হয় হলুদ হবে(বর্ণনা অনুযায়ী ) ।

শাফায়েত

২৭ ডিসেম্বর ২০১৩

না সবুজই হবে। “যেসব নোডে এসে অন্তত একটা শব্দ কমপ্লিট হয়েছে সেসব নোডে আমরা এন্ডমার্ক বসিয়ে দেই, ছবিতে সবুজ রঙ হলো এন্ডমার্ক।” শেষে আমরা “to” যোগ করসি তাই o এজ এর নিচের নোড সবুজ হবে। কোন লাইনে তোমার কনফিউশন হচ্ছে জানাও।

সান

২৭ ডিসেম্বর ২০১৩

দুঃখিত ভাইয়া , আমার বুঝার ভুল ছিল ।

ধন্যবাদ ভাইয়া , এত সহজে বুঝানোর জন্য ।

অনিন্দ্য

৩১ জুলাই ২০১৪

ভাইয়া, ট্রাইই ইমপ্লিমেন্ট করলে সেটা কি মেমরি লিমিটের মধ্যে রান করবে নাকি এটা হিসাব করবো কিভাবে?

মন্তব্য করুন(বাংলা ভাষায় বাংলা ফন্টে বা ইংরেজি ভাষায় ইংরেজি ফন্টে)

নাম(আবশ্যক)

ইমেইল(আবশ্যক)

ওয়েবসাইট

1 + = 2

(ইংরেজী ফন্টে বাংলা মন্তব্য মুছে ফেলা হতে পারে)

[লোয়েস্ট কমন অ্যানসেস্টর](http://www.shafaetsplanet.com/planetcoding/?p=1831)

লেখক: শাফায়েত তারিখ: মার্চ ১৩, ২০১৪ বিভাগ: [কনটেস্ট প্রোগ্রামিং/প্রবলেম সলভিং](http://www.shafaetsplanet.com/planetcoding/?cat=35), [প্রোগ্রামিং](http://www.shafaetsplanet.com/planetcoding/?cat=1)

ট্যাগ: [ট্রি](http://www.shafaetsplanet.com/planetcoding/?tag=%e0%a6%9f%e0%a7%8d%e0%a6%b0%e0%a6%bf)• [ডাটা স্ট্রাকচার](http://www.shafaetsplanet.com/planetcoding/?tag=%e0%a6%a1%e0%a6%be%e0%a6%9f%e0%a6%be-%e0%a6%b8%e0%a7%8d%e0%a6%9f%e0%a7%8d%e0%a6%b0%e0%a6%be%e0%a6%95%e0%a6%9a%e0%a6%be%e0%a6%b0)• [লোয়েস্ট কমন অ্যানসেস্টর](http://www.shafaetsplanet.com/planetcoding/?tag=%e0%a6%b2%e0%a7%87%e0%a6%be%e0%a7%9f%e0%a7%87%e0%a6%b8%e0%a7%8d%e0%a6%9f-%e0%a6%95%e0%a6%ae%e0%a6%a8-%e0%a6%85%e0%a7%8d%e0%a6%af%e0%a6%be%e0%a6%a8%e0%a6%b8%e0%a7%87%e0%a6%b8%e0%a7%8d%e0%a6%9f%e0%a6%b0)• [স্পার্স টেবিল](http://www.shafaetsplanet.com/planetcoding/?tag=%e0%a6%b8%e0%a7%8d%e0%a6%aa%e0%a6%be%e0%a6%b0%e0%a7%8d%e0%a6%b8-%e0%a6%9f%e0%a7%87%e0%a6%ac%e0%a6%bf%e0%a6%b2)

লোয়েস্ট কমন অ্যানসেস্টর জিনিসটা শুনতে একটু কঠিন মনে হলেও জিনিসটা সহজ আর খুবই কাজের। বেশ কিছু ধরণের প্রবলেম সলভ করে ফেলা যায় এই অ্যালগোরিদম দিয়ে। আমরা প্রথমে লোয়েস্ট কমন অ্যানসেস্টর বের করার ব্রুটফোর্স অ্যালগোরিদম দেখবো, তারপর স্পার্স টেবিল নামের নতুন একটা ডাটা স্ট্রাকচার শিখে কমপ্লেক্সিটি লগ এ নামিয়ে আনবো।

প্রথমেই আমরা জেনে নেই লোয়েস্ট কমন অ্যানসেস্টর বা এল.সি.এ(LCA) কি সেটা। নিচের ছবিটা দেখ:

ছবিতে k আর n নোডের প্যারেন্ট ধরে পিছে যেতে থাকলে তারা i নোডে এসে মিলবে। i হলো k,n এর লোয়েস্ট কমন অ্যানসেস্টর। ‘a’ ও দুইজনের কমন অ্যানসেস্টর কিন্তু i হলো ‘লোয়েস্ট’ বা সবথেকে কাছাকাছি।

মনে করো ট্রি টা দিয়ে বুঝানো হচ্ছে একটা অফিসে কে কার সুপারভাইজর বা বস। a হলো অফিসের হেড, b এবং i এর বস হলো a, আবারc এবং f এর বস হলো b ইত্যাদি। তাহলে লোয়েস্ট কমন অ্যানসেস্টর অ্যালগোরিদম দিয়ে দুইজন এমপ্লয়ির লোয়েস্ট/সব থেকে কাছের কমন বস খুজে বের করে ফেলতে পারি সহজেই।

প্রথমে আমরা একদম নেইভ একটা উপায় দেখি যেটা খুব বড় ইনপুটের জন্য কাজ করবেনা। মনে করি আমরা k,n এর এল.সি.এ বের করতে চাই, প্রথমেই k এর প্যারেন্ট ধরে ধরে উপরে উঠে সবগুলো অ্যানসেস্টরের লিস্ট করে ফেলি:

**k এর অ্যানসেস্টর:**

|  |  |  |
| --- | --- | --- |
| j | i | a |

**n এর অ্যানসেস্টর:**

|  |  |  |
| --- | --- | --- |
| m | i | a |

দুইটা লিস্টেই বাম থেকে ডানে যেতে থাকলে প্রথম যে দুইটা নোড কমন পাবো সেটাই হবে এল.সি.এ! এই উদাহরণে প্রথমে কমন পাবো i,তাই সেটাই হবে এল.সি.এ। ‘a’ ও একটা কমন এনসেস্টর কিন্তু সেটা “লোয়েস্ট” না। একটা ভিজিটেড অ্যারে বা ফ্ল্যাগ ব্যবহার করে খুব সহজেই আমরা এই অ্যালগোরিদম ইমপ্লিমেন্ট করতে পারি। প্রথমে k এর লিস্টে যারা আছে সেগুলোর ইনডেক্সের ফ্ল্যাগ অন করে দিবো,তারপর n এর লিস্টে লুপ চালিয়ে দেখবো ফ্ল্যাগ অন আছে কোন নোডের।

প্রতিটা লিস্ট বানাতে আমাদের সর্বোচ্চ n টা নোড ভিজিট করা লাগতে পারে, তাই কমপ্লেক্সিটি O(n) ।

O(n) কমপ্লিক্সিটি খারাপ না যদি আমাদের মাত্র এক জোড়া নোডের এল.সি.এ বের করতে হয়। যদি আমাদের m জোড়া নোড দিয়ে বলে প্রতিটা পেয়ারের এল.সি.এ বের করতে তাহলে কমপ্লেক্সিটি হয়ে যাবে O(m\*n) । এটাকে আমরা চাইলে O(m\*log(n)) এ সলভ করতে পারি, অর্থাৎ প্রতি কুয়েরি কাজ করবে O(logn) এ। তবে তার আগে কিছু প্রি-প্রসেসিং করে নিতে হবে।

প্রতিটা নোডের জন্য আমরা সেভ করে রাখবো নোডটার:

* প্রথম প্যারেন্ট বা ২^০ তম প্যারেন্ট
* ২য় প্যারেন্ট বা ২^১ তম প্যারেন্ট
* ৪র্থ প্যারেন্ট বা ২^২ তম প্যারেন্ট
* ৮ম প্যারেন্ট বা ২^৩ তম প্যারেন্ট

যদি K-তম প্যারেন্ট বলতে কিছু না থাকে তাহলে আমরা -১ রেখে দিবো। যেমন রুট নোডের প্রথম প্যারেন্ট বলতে কিছু নেই, তাই প্রথম প্যারেন্ট হবে -১।

তাহলে উপরের ট্রি এর জন্য নিচের মতো একটা ২-ডিমেনশনাল টেবিল তৈরি হবে:

|  |  |  |  |
| --- | --- | --- | --- |
| **নোড** | **২^০ তম প্যারেন্ট** | **২^১ তম প্যারেন্ট** | **২^২ তম প্যারেন্ট** |
| ০ | -১ | -১ | -১ |
| ১ | ০ | -১ | -১ |
| ২ | ০ | -১ | -১ |
| ৩ | ০ | -১ | -১ |
| ৪ | ২ | ০ | -১ |
| ৫ | ৪ | ২ | -১ |
| ৬ | ৫ | ৪ | ০ |
| ৭ | ৫ | ৪ | ০ |
| ৮ | ০ | -১ | -১ |

তাহলে উপরের ট্রি এর জন্য নিচের মতো একটা ২-ডিমেনশনাল টেবিল তৈরি হবে:

এই টেবিলটাকে বলে **স্পার্স টেবিল।** এই টেবিলটা তৈরি করে কি লাভ হলো? এখন তুমি কোনো নোডের k তম প্যারেন্ট খুব সহজেই খুজে বের করতে পারবে। যেমন কারো ২৫ তম প্যারেন্ট দরকার হলে প্রথমে ২৫ এর নিচে ২ এর সবথেকে বড় পাওয়ার ২^৪=১৬ তম প্যারেন্ট খুজে বের করবে। তারপর ১৬তম প্যারেন্টের (২৫-১৬)=৯তম প্যারেন্ট খুজে বের করবে একই পদ্ধতিতে! প্রতিটা সংখ্যাকে ২ এর পাওয়ারের যোগফল হিসাবে প্রকাশ করা যায় সেটারই সুবিধা নিচ্ছি আমরা। এভাবে করে O(logn) কমপ্লেক্সিটিতে আমরা কারো k তম প্যারেন্ট বের করতে পারবো যেটা পরবর্তীতে LCA বের করতে কাজে লাগবে।

এখন কথা হলো উপরের টেবিলটা বানাবো কিভাবে? ২^০ তম প্যারেন্ট বের করা সহজ, ট্রি এর উপর একটা ডেপথ-ফার্স্ট-সার্চ বা ব্রেথড-ফার্স্ট-সার্চ চালিয়ে একটা অ্যারেতে সেভ করে রাখবো কার প্যারেন্ট কে। মনে করি অ্যারেটা হলো T। তাহলে উপরের ট্রি এর জন্য পাবোT[০]=-১, T[১]=০,T[৬]=৫ ইত্যাদি।

এবার আমরা কলাম-বাই-কলাম টেবিলটা ভরাট করবো। প্রথমেই প্রথম কলাম ভরাট করবো যেটা আসলে T অ্যারের সমান।

টেবিলটা নাম P হলে প্রথম কলাম ভরাট করবো এভাবে,



|  |  |
| --- | --- |
| 1  2 | for (i = 0; i < N; i++)           P[i][0] = T[i]; |

এখন চিন্তা করে দেখো একটা নোডের ২^৪=১৬ তম প্যারেন্ট হলো নোডটার ২^৩=৮ম প্যারেন্টের ২^৩=৮ম প্যারেন্ট। তাহলে কোন নোডের 2^j তম প্যারেন্ট হবে নোডটার 2^(j-1) তম প্যারেন্টের 2^(j-1) তম প্যারেন্ট! তাহলে যেকোন নোড i এর জন্য:

P[i][j]=P[P[i][j - 1]][j – 1] যেখানে P[i][j – 1] হলো j-1 তম প্যারেন্ট।

তাহলে পুরো টেবিলটা ভরাট করে ফেলতে পারি এভাবে:



|  |  |
| --- | --- |
| 1  2  3  4 | for (j = 1; (1 << j) < N; j++)          for (i = 0; i < N; i++)                if (P[i][j - 1] != -1)                  P[i][j] = P[P[i][j - 1]][j – 1]; |

2^j এর মান যদি N এর ছোট না হয় তাহলে নতুন কোনো প্যারেন্ট পাবার সম্ভাবনা নেই, তাই লুপ চলবে (1<<j) বা 2^j যতক্ষণ N এর ছোট হবে।

এখন আমরা দুটি নোডের এল.সি.এ বের করবো। আমাদের আরেকটা অতিরিক্ত অ্যারে লাগবে **L[]** যেখানে থাকবে প্রতিটা নোডের ডেপথ বা লেভেল। **রুট নোডের লেভেল হবে ০।**

এখন আমরা হলুদ নোড দুইটার এল.সি.এ বের করতে চাই:

প্রথম কাজ হবে এদেরকে একই লেভেলে নিয়ে আসা। p এর লেভেল থেকে ২ এর পাওয়ার বিয়োগ করতে থাকবো এবং একই সাথে স্পার্স টেবিল ব্যবহার করে উপরে উঠাতে থাকবো যতক্ষণনা q এর লেভেলের সমান হয়।



|  |  |
| --- | --- |
| 1  2  3 | for (i = log; i >= 0; i--)            if (L[p] - (1 << i) >= L[q]) //2^i তম প্যারেন্টে যাও                p = P[p][i]; |

এখন দুইজন সমান লেভেলে আসলো। এবার কাজ হবে দুইটা নোডকেও টেনে একসাথে উপরে উঠানো যতক্ষনা নোড দুটির প্যারেন্ট একসাথে এসে মিলে:

কোডটা হবে এরকম:



|  |  |
| --- | --- |
| 1  2  3 | for (i = log; i >= 0; i--)            if (P[p][i] != -1 && P[p][i] != P[q][i])                p = P[p][i], q = P[q][i]; |

আমরা এমন জায়গায় p,q কে আনছি যেখানে তাদের দুইজনেরই প্যারেন্ট একই। তাহলে এখন p বা q এর প্যারেন্টই হবে এল.সি.এ।



|  |  |
| --- | --- |
| 1 | return T[p]; |

আমরা যদি লুপের মধ্যেই ২ আর ৮ নাম্বার নোডকে p,q কে না এনে ডিরেক্ট ১ নম্বর নোডে আনার চেষ্টা করতাম তাহলে কি ঘটতে পারতো?এটা চিন্তা করা তোমার কাজ :) ।

**কমপ্লেক্সিটি:**

স্পার্স টেবিলে রো থাকবে n টা, প্রতিটা রো তে logn টা কলাম থাকবে, প্রি-প্রোসেসিং কমপ্লেক্সিটি O(nlogn)।

প্রতি কুয়েরিতে কমপ্লেক্সিটি O(logn)

**রিলেটেড প্রবলেম:**

১. একটা ওয়েটেড ট্রি দেয়া আছে। দুটি নোড দিয়ে বলা হলো তাদের মধ্যের দূরত্ব বের করতে হবে। [(Spoj QTree)](http://www.spoj.com/problems/QTREE2/)

২. একটা ওয়েটেড ট্রি দেয়া আছে। দুটি নোড দিয়ে বলা হলো তাদের মধ্যকার পাথের সর্বোচ্চ ওয়েটের এজ এর মান বলতে হবে। [(LOJ A Secret Mission)](http://lightoj.com/volume_showproblem.php?problem=1101)

সলিউশন ১: শুরুতে ডিএফস চালিয়ে রুট থেকে প্রতিটা নোডের দূরত্ব সেভ করে রাখো। এখন নোড দুইটা a,b এবং তাদের কমন অ্যানসেস্টর L হলে সলিউশন হবে dist(root,a)+dist(root,b)-2\*dist(root,L)।

সলিউশন ২: স্পার্স টেবিলে অতিরিক্ত একটা ইনফরমেশন রাখতে হবে।  প্রতিটা নোড থেকে 2^0, 2^1, 2^2 ইত্যাদি তম প্যারেন্টের জন্য, প্যারেন্ট থেকে ওই নোডের মধ্যকার পাথের সর্বোচ্চ ওয়েটের এজের মান সেভ করে রাখতে হবে। বিস্তারিত লিখলাম, এটা সলভ করা তোমার কাজ।

**আরো কিছু প্রবলেম:**

[TJU Closest Common Ancestors  
UVA Flea Circus](http://uva.onlinejudge.org/external/109/10938.html)  
[LightOJ LCA Category](http://lightoj.com/volume_problemcategory.php?category=Range%20Minimum%20Query/Lowest%20Common%20Ancestor)

**রিসোর্স:**

এল.সি.এ বের করার আরো কিছু পদ্ধতি আছে যেগুলো পাবে [এখানে](http://community.topcoder.com/tc?module=Static&d1=tutorials&d2=lowestCommonAncestor)। এগুলো ছাড়াও টারজানের একটা [অফলাইন অ্যালগোরিদম](http://en.wikipedia.org/wiki/Tarjan%27s_off-line_lowest_common_ancestors_algorithm) আছে যেটা আগে সব কুয়েরি ইনপুট নিয়ে একটি ডিএফস চালিয়ে সবগুলো পেয়ারের এল.সি.এ বের করে দেয়।

**সম্পূর্ণ কোড:**

ভেক্টর g তে ট্রি সেভ করতে হবে। প্রথমে dfs ফাংশন কল করে লেভেল,প্যারেন্ট ফিক্সড করে তারপর lca\_init কল করে প্রিপ্রসেসিং করতে হবে।

|  |  |
| --- | --- |
| 1234567891011121314151617181920212223242526272829303132333435363738394041424344454647484950515253545556575859606162636465666768697071 | //LCA using sparse table  //Complexity: O(NlgN,lgN)  #define mx 100002  int L[mx]; //লেভেল  int P[mx][22]; //স্পার্স টেবিল  int T[mx]; //প্যারেন্ট  vector<int>g[mx];  void dfs(int from,int u,int dep)  {  T[u]=from;  L[u]=dep;  for(int i=0;i<(int)g[u].size();i++)  {  int v=g[u][i];  if(v==from) continue;  dfs(u,v,dep+1);  }  }    int lca\_query(int N, int p, int q) //N=নোড সংখ্যা  {  int tmp, log, i;    if (L[p] < L[q])  tmp = p, p = q, q = tmp;    log=1;  while(1) {  int next=log+1;  if((1<<next)>L[p])break;  log++;    }    for (i = log; i >= 0; i--)  if (L[p] - (1 << i) >= L[q])  p = P[p][i];    if (p == q)  return p;    for (i = log; i >= 0; i--)  if (P[p][i] != -1 && P[p][i] != P[q][i])  p = P[p][i], q = P[q][i];    return T[p];  }    void lca\_init(int N)  {  memset (P,-1,sizeof(P)); //শুরুতে সবগুলো ঘরে -১ থাকবে  int i, j;  for (i = 0; i < N; i++)  P[i][0] = T[i];    for (j = 1; 1 << j < N; j++)  for (i = 0; i < N; i++)  if (P[i][j - 1] != -1)  P[i][j] = P[P[i][j - 1]][j - 1];  }    int main(void) {  g[0].pb(1);  g[0].pb(2);  g[2].pb(3);  g[2].pb(4);  dfs(0, 0, 0);  lca\_init(5);  printf( "%d\n", lca\_query(5,3,4) );  return 0;  } |

[**view raw**](https://gist.github.com/Shafaet/9757472/raw/LCA)[**LCA**](https://gist.github.com/Shafaet/9757472#file-lca) hosted with ❤ by [**GitHub**](https://github.com/)

হ্যাপি কোডিং!

[Print Friendly](http://www.printfriendly.com/print?url=http://www.shafaetsplanet.com/planetcoding/?p=1831)

ফেসবুকে মন্তব্য

7 comments

Powered by [Facebook Comments](http://peadig.com/wordpress-plugins/facebook-comments/)

ট্যাগ: [ট্রি](http://www.shafaetsplanet.com/planetcoding/?tag=%e0%a6%9f%e0%a7%8d%e0%a6%b0%e0%a6%bf)• [ডাটা স্ট্রাকচার](http://www.shafaetsplanet.com/planetcoding/?tag=%e0%a6%a1%e0%a6%be%e0%a6%9f%e0%a6%be-%e0%a6%b8%e0%a7%8d%e0%a6%9f%e0%a7%8d%e0%a6%b0%e0%a6%be%e0%a6%95%e0%a6%9a%e0%a6%be%e0%a6%b0)• [লোয়েস্ট কমন অ্যানসেস্টর](http://www.shafaetsplanet.com/planetcoding/?tag=%e0%a6%b2%e0%a7%87%e0%a6%be%e0%a7%9f%e0%a7%87%e0%a6%b8%e0%a7%8d%e0%a6%9f-%e0%a6%95%e0%a6%ae%e0%a6%a8-%e0%a6%85%e0%a7%8d%e0%a6%af%e0%a6%be%e0%a6%a8%e0%a6%b8%e0%a7%87%e0%a6%b8%e0%a7%8d%e0%a6%9f%e0%a6%b0)• [স্পার্স টেবিল](http://www.shafaetsplanet.com/planetcoding/?tag=%e0%a6%b8%e0%a7%8d%e0%a6%aa%e0%a6%be%e0%a6%b0%e0%a7%8d%e0%a6%b8-%e0%a6%9f%e0%a7%87%e0%a6%ac%e0%a6%bf%e0%a6%b2)

Nikhil Haldipuri

৩১ জুলাই ২০১৪

আমিএকটা জিনিস বুঝতে পার্ছি না LCA query function এতে N কি কাজে লাগ্ছে?

শাফায়েত

৩১ জুলাই ২০১৪

কাজে লাগছে না, বাদ দিয়ে দিতে পারো।

Nikhil Haldipuri

৩১ জুলাই ২০১৪

ধন্যবাদ ভাই

মন্তব্য করুন(বাংলা ভাষায় বাংলা ফন্টে বা ইংরেজি ভাষায় ইংরেজি ফন্টে)

নাম(আবশ্যক)

ইমেইল(আবশ্যক)

ওয়েবসাইট

+ 2 = 3

(ইংরেজী ফন্টে বাংলা মন্তব্য মুছে ফেলা হতে পারে)

[ডাটা স্ট্রাকচার: বাইনারি ইনডেক্সড ট্রি](http://www.shafaetsplanet.com/planetcoding/?p=1961)

লেখক: শাফায়েত তারিখ: আগস্ট ১৯, ২০১৪ বিভাগ: [কনটেস্ট প্রোগ্রামিং/প্রবলেম সলভিং](http://www.shafaetsplanet.com/planetcoding/?cat=35), [প্রোগ্রামিং](http://www.shafaetsplanet.com/planetcoding/?cat=1)

ট্যাগ: [ডাটা স্ট্রাকচার](http://www.shafaetsplanet.com/planetcoding/?tag=%e0%a6%a1%e0%a6%be%e0%a6%9f%e0%a6%be-%e0%a6%b8%e0%a7%8d%e0%a6%9f%e0%a7%8d%e0%a6%b0%e0%a6%be%e0%a6%95%e0%a6%9a%e0%a6%be%e0%a6%b0)• [বাইনারি ইনডেক্সড ট্রি](http://www.shafaetsplanet.com/planetcoding/?tag=%e0%a6%ac%e0%a6%be%e0%a6%87%e0%a6%a8%e0%a6%be%e0%a6%b0%e0%a6%bf-%e0%a6%87%e0%a6%a8%e0%a6%a1%e0%a7%87%e0%a6%95%e0%a7%8d%e0%a6%b8%e0%a6%a1-%e0%a6%9f%e0%a7%8d%e0%a6%b0%e0%a6%bf)

বাইনারি ইনডেক্স ট্রি খুবই চমৎকার একটা [ডাটা স্ট্রাকচার](http://www.shafaetsplanet.com/planetcoding/?tag=%E0%A6%A1%E0%A6%BE%E0%A6%9F%E0%A6%BE-%E0%A6%B8%E0%A7%8D%E0%A6%9F%E0%A7%8D%E0%A6%B0%E0%A6%BE%E0%A6%95%E0%A6%9A%E0%A6%BE%E0%A6%B0) যার সবথেকে ভালো দিক হলো কয়েক মিনিটেই কোড লিখে ফেলা যায়। আর খারাপ দিক হলো ভিতরে কি হচ্ছে বুঝতে একটু কষ্ট হয়, তবে তুমি লেখাটা মনযোগ দিয়ে পড়লে আমি নিশ্চিত কোন সমস‍্যা হবে না। বাইনারি ইনডেক্সড ট্রি বা BIT এর আরেক নাম হলো ফেনউইক(fenwick) ট্রি। এই লেখাটা পড়ার আগে তোমাকে বিটওয়াইজ অপারেশনগুলো(AND,OR ইত‍্যাদি) সম্পর্কে ভালো জানতে হবে।

আমরা একটা সহজ সমস‍্যা সমাধান করতে করতে বাইনারি ইনডেক্স ট্রি সম্পর্কে জানব। এই লেখায় আমরা ধরে নিব অ‍্যারের ইনডেক্স ১ থেকে শুরু, BIT কিভাবে কাজ করে জানার পরে বুঝতে পারবে এটা কেন গুরুত্বপূর্ণ। তোমাকে একটা অ‍্যারে দেয়া হলো যার প্রতিটি পজিশনে ০ আছে। এখন তোমাকে অনেকগুলো অপারেশন দেয়া হবে। প্রতিটা অপারেশন হতে পারে নিচের যেকোন একটা:

১. i তম ইনডেক্সে x স‍ংখ‍্যাটা যোগ কর।  
২. 1 থেকে i তম ইনডেক্সের সাবঅ‍্যারের মোট যোগফল বল।

একদম ‘নেইভ(Naive)’ উপায়ে করলে আমরা O(1) এ প্রতি পজিশনে ১ যোগ করব আর লুপ চালিয়ে O(n) এ যোগফল বের করব। BIT দিয়ে যোগফল O(logn) এ বের করা সম্ভব, তবে সেক্ষেত্রে আপডেট অপারেশনও O(logn) এ করতে হবে।  
আমরা জানি প্রতিটা সংখ‍্যাকেই কিছু ২ এর পাওয়ারের যোগফল হিসাবে লেখা যায়। যেমন ১৩ এর বাইনারি হলো “১১০১”, বাইনারিতে ১ আছে ০তম,২য় আর ৩য় অবস্থানে। তাহলে ১৩ কে লেখা যায় ২^০+২^২+২^৩ বা ১+৪+৮। বাইনারি ইনডেক্স ট্রি তে আমরা এই প্রোপার্টিটা ব‍্যবহার করে কিছু কাজ করব। যে সংখ‍্যাটা অ‍্যারেতে যোগ করছি সেটাকে নয় বরং আমরা অ‍্যারের ইনডেক্সগুলোকে ২ এর পাওয়ারে যোগফল হিসাবে লিখে কিছু কাজ করব।

1 থেকে i পর্যন্ত অ‍্যারের সামকে আমরা অ‍্যারের কিছু সেগমেন্টের যোগফল হিসাবে লিখতে পারি। মনে কর আমাদের একটা sum(i,j) ফাংশন আছে যার কাজ হলো i থেকে j ইনডেক্সের মধ‍্যে অ‍্যারের সবগুলো এলিমেন্টের যোগফল বের করা। তাহলে ১৩ তম ইনডেক্সের এর জন‍্য যোগফল হতে পারে sum(1,5)+sum(6,10)+sum(11,12)। এখানে ১৩কে আমরা ৩টি সেগমেন্টে ভাগ করেছি। বাইনারি ইনডেক্স ট্রি তে আমরা প্রতিটি সংখ‍্যাকে কিছু সেগমেন্টে ভাগ করব। তবে উপরে যেভাবে করেছি সেভাবে ইচ্ছামত করলে হবে না, এটা আমরা একটু বুদ্ধিমানের মত করব। যেহেতু “বাইনারি ইনডেক্সড ট্রি” ব‍্যবহার করব, তাই বুঝতে পারছ হয়ত যে অ‍্যারের ইনডেক্সগুলোর বাইনারি রিপ্রেজেন্টশনকে ব‍্যবহার করে কোন কাজ করতে হবে।

কোন সংখ‍্যাকে যেহেতু ২এর পাওয়ারের যোগফল হিসাবে লেখা যায়, তারমানে সংখ‍্যাটা থেকে ২এর পাওয়ার বিয়োগ দিতে দিতে আমরা ০ বানিয়ে ফেলতে পারি। ১৩ থেকে একে একে ১,৪ এবং ৮ বাদ দিলে আমরা পাবো:

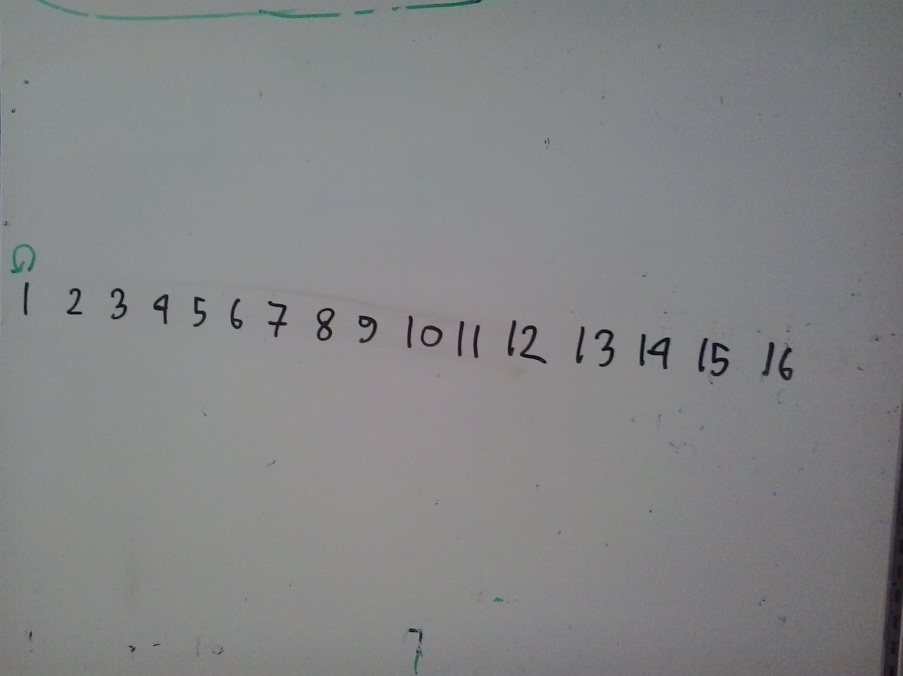
১৩-১=১২  
১২-৪=৮  
৮-৮=০

তাহলে ১৩ এর জন‍্য সেগমেন্ট হবে sum(1,8) + sum(9,12) + sum(13,13)। আরেকটা উদাহরণ দেখলে পরিস্কার হবে, ৬(বাইনারিতে ১১০)=২^১+২^২ এর জন‍্য একই ভাবে লিখতে পারি:

৬-২=৪  
৪-৪=০

এবার তাহলে ৬ এর জন‍্য সেগমেন্ট হবে sum(1,4)+sum(5,6)।

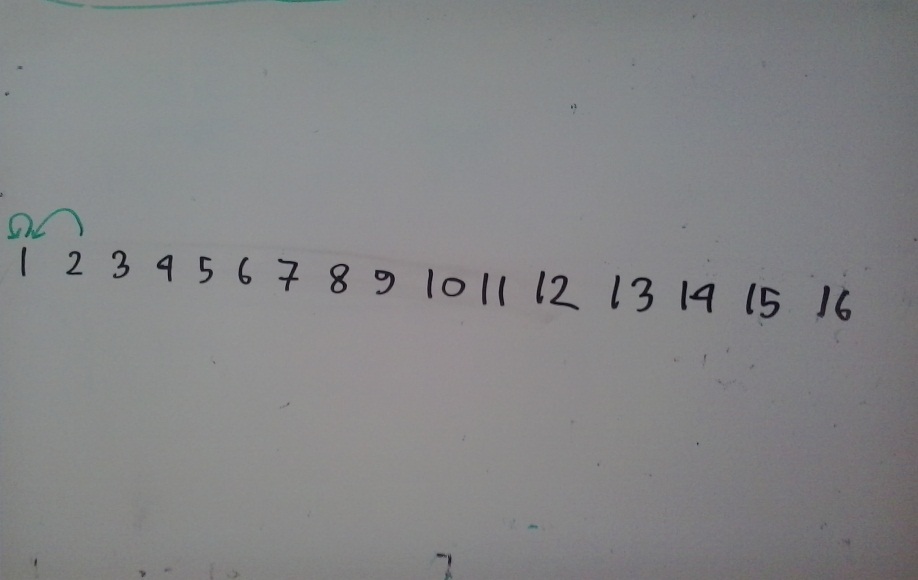
কোড কিভাবে করব সেটা এখন চিন্তা করা দরকার নেই, তুমি শুধু চিন্তা কর গাণিতিকভাবে ভাগটা কিভাবে করা হচ্ছে সেটা তুমি বুঝতে পারছ নাকি। এখন আমরা সেগমেন্টগুলো ছবিতে দেখব:

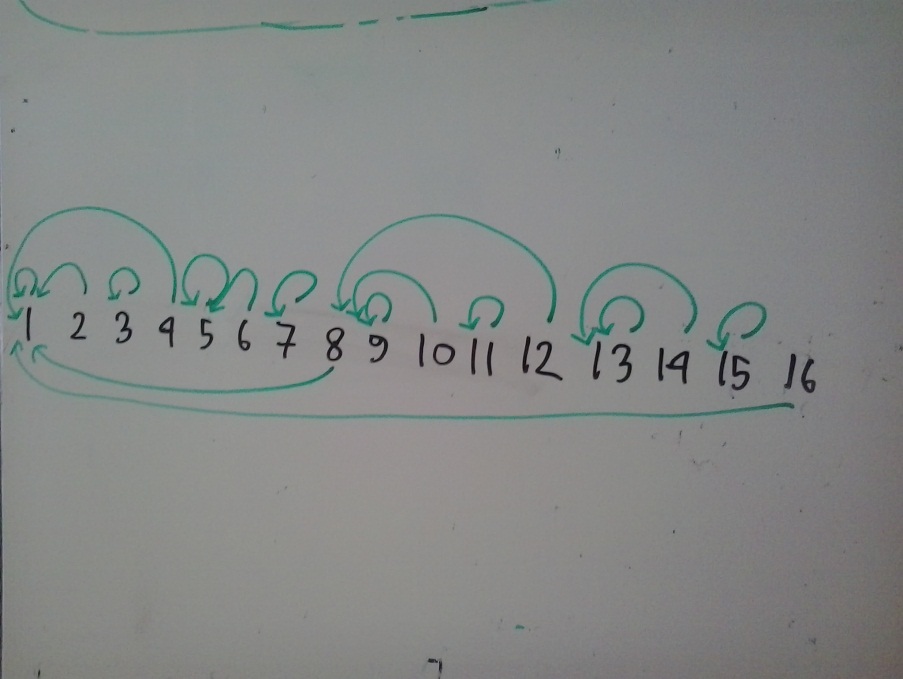
[](http://www.shafaetsplanet.com/planetcoding/wp-content/uploads/2014/08/2014-08-19-12.18.06.jpg)

ছবিতে ১ এর জন‍্য সেগমেন্টটা চিহ্নিত করা হয়েছে। এরপরে আমরা ২এর জন‍্য করব। ২ এর জন‍্য উপরের মত করে লিখতে পারি:

২-২=০

তাই সেগমেন্টটা হবে ১থেকে ২পর্যন্ত।

[](http://www.shafaetsplanet.com/planetcoding/wp-content/uploads/2014/08/2014-08-19-12.18.44.jpg)এখন তোমার কাজ ১৬ পর্যন্ত সবগুলো সংখ‍্যার জন‍্য এভাবে সেগমেন্ট একে ফেলা। কিভাবে সেগমেন্ট তৈরি হচ্ছে বোঝাটা বাইনারি ইনডেক্সড ট্রি এর সবথেকে দরকারি অংশ, তাই না বুঝলে আবার ভাল করে পড়। তোমার আকার সাথে নিচের ছবিটি মিলিয়ে দেখ:

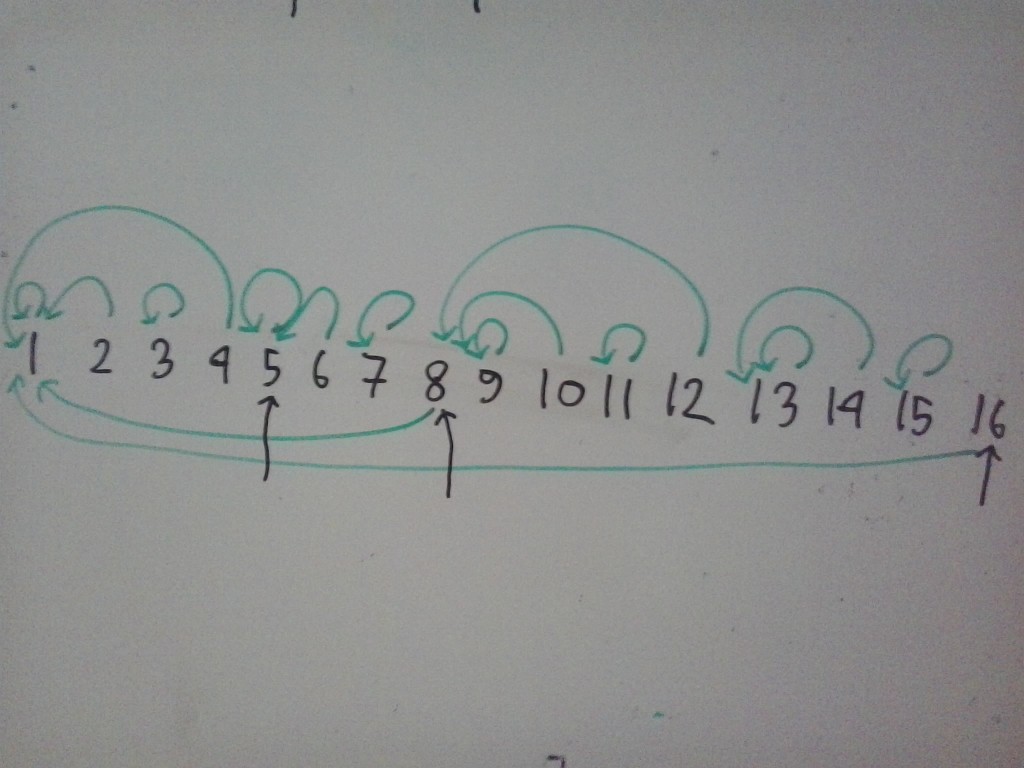
[](http://www.shafaetsplanet.com/planetcoding/wp-content/uploads/2014/08/2014-08-19-12.22.10.jpg)

এই ছবি দেখেই আমরা প্রতিটা সংখ‍্যার জন‍্য সেগমেন্টগুলো বলে দিতে পারি। যেমন ১১ এর জন‍্য সেগমেন্টগুলো হবে (11,11), (10,9), (8,1)। তুমি যদি নিজে বুঝে উপরের ছবিটা একে থাকো তাহলে তুমি বুঝে গেছ কিভাবে ছবি দেখে সেগমেন্ট বের করে ফেললাম।

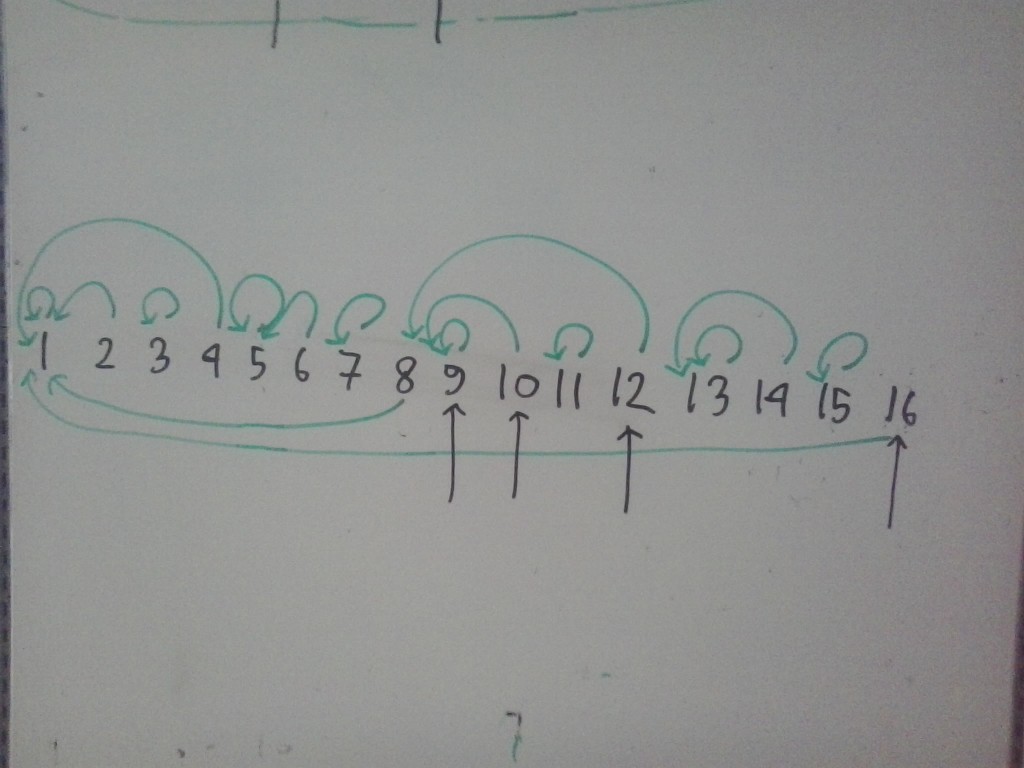
আমরা আগেই sum(i,j) ফাংশন ডিফাইন করেছি। কিন্তু সেই ফাংশনটা ব‍্যবহার করলে আমাদেরকে i থেকে j তে লুপ চালিয়ে যোগফল বের করতে হবে যেটা আমরা চাইনা। আমরা আরেকটা অ‍্যারে ডিফাইন করব Tree[]। তুমি ছবিতে দেখতে পাচ্ছ অ‍্যারের প্রতিটি ইনডেক্স একটা সাবঅ‍্যারেকে কভার করে। যেমন ১২ নম্বর ইনডেক্স (৮,১২) সাবঅ‍্যারেকে কভার করে। আমরা এমন কিছু করার চেষ্টা করব যেন Tree[i] তে i তম ইনডেক্স যতটুকু সাবঅ‍্যারে কভার করে সেটার যোগফল থাকে। তাহলে Tree[12] তে সেভ করা থাকবে sum(8,12), আবার Tree[14] তে সেভ করা থাকবে sum(13,14)।

এই কাজটা যদি করতে পারি তাহলে ১১ নম্বর ইনডেক্সের জন‍্য sum(11,11)+sum(10,9)+sum(8,1) বের না করে আমরা tree[11]+tree[10]+tree[8] বের করলেই যোগফল পেয়ে যাবো।

এখন ধরি আমাকে বলা হল ৫ নম্বর ইনডেক্সে x যোগ করতে চাই। আমাকে এখন বুদ্ধিমানের মত Tree অ‍্যারেতে কিছু ইনডেক্সে x সং‍খ‍্যাটা যোগ করে দিতে হবে। আমরা প্রথমে Tree[5] এ x যোগ করব, এরপরে সবথেকে ছোট সেই ইনডেক্সে করব যেটা উপরের ছবিতে ৫কে কভার করে। সেই ইনডেক্সটা হলো ৮, তাহলে tree[8] এ x যোগ করব। এরপরে ৮কে কভার করে ১৬, তাই tree[16] এ x যোগ করব।

[](http://www.shafaetsplanet.com/planetcoding/wp-content/uploads/2014/08/2014-08-19-17.08.10.jpg)

আমরা tree অ‍্যারেতে ছবিতে তীর চিহ্ন দেয়া ইনডেক্সগুলোতে x যোগ করব। এখানে লক্ষ‍্য করার বিষয় হলো উপরে লেখা পদ্ধতিতে যোগফল বের করার সময় তুমি এই ৩টা ইনডেক্সের সর্বোচ্চ যেকোন এক্সেস করতে পারবে। যেমন ১৬ তে এক্সেস করলে সরাসরি ৫,৮ ইত‍্যাদিকে স্কিপ করে সরাসরি ১ এ চলে যেতে হবে। এই প্রোপার্টিটা থাকার কারণে একই আপডেট দুইবার গুণে ফেলার সম্ভাবনা নেই। আপডেট অপারেশন ঠিকমত বুঝেছো নাকি পরীক্ষা করতে ৯ এর জন‍্য উপরের মত করে তীরচিহ্ন বসায় এবং তারপর নিচের সমাধানের সাথে মিলিয়ে দেখ:

[](http://www.shafaetsplanet.com/planetcoding/wp-content/uploads/2014/08/2014-08-19-17.29.35.jpg)

ছবিতে ৯থেকে শুরু করে প্রতিবার এমন ইনডেক্সে গিয়েছি যেটা আগের ইনডেক্সটাকে পুরোটা কভার করে। যোগফল বের করার সময় কোনভাবেই এই ৪টা ইনডেক্সের একটার বেশি এক্সেস করা সম্ভব না!

এবার আমরা কোডের দিকে যাবো। প্রথমেই কুয়েরি অপারেশনটা ইমপ্লিমেন্ট করব। কোন সংখ‍্যা ২ এর পাওয়ার হলে তার বাইনারি রিপ্রেজেন্টশন হয় ১, ১০,১০০,১০০০,১০০..০০ এরকম, অর্থাৎ ১ এর পর কিছু শূন‍্য। এখন ১১ নাম্বার ইনডেক্সের জন‍্য কুয়েরি করতে চাই। ১১ কে বাইনারিতে লিখতে পারি ১০১১। আমরা প্রথমেই সবথেকে ডানের ১ খুজে বের করব এবং সেই পজিশন থেকে বাকি অংশটা মূল সংখ‍্যা থেকে বিয়োগ দিব। তাহলে শুরুতেই বিয়োগ হবে ১।

১০১১-১=১০১০ (অথবা দশমিক পদ্ধতিতে ১১-১=১০)

আবারো সব থেকে ডানের ১ থেকে শুরু করে বাকিটা বিয়োগ দিলে পাব:

১০১০-১০=১০০০ (অথবা দশমিক পদ্ধতিতে ১০-২=৮)

এভাবে বিয়োগ করতে থাকবো যতক্ষণ শূন‍্য না পাই।

১০০০-১০০০=০ (অথবা দশমিক পদ্ধতিতে ৮-৮=০)

লক্ষ‍্য কর ১১ এর জন‍্য দরকারি ৩টা ইনডেক্স ১১,১০,৮ আমরা পেয়ে গিয়েছি ২এর পাওয়ার বিয়োগ করে করে! ১৩ বা বাইনারিতে ১১০১ এর জন‍্য নিজে কর এবং তারপর নিচের সমাধান দেখ:

১১০১-১=১১০০ (অথবা দশমিক পদ্ধতিতে ১৩-১=১২)

১১০০-১০০=১০০০ (অথবা দশমিক পদ্ধতিতে ১২-৪=৮)

১০০০-১০০০=০   (অথবা দশমিক পদ্ধতিতে ৮-৮=০)

১৩ এর জন‍্য দরকারি ইনডেক্স ১৩,১২,৮ পেয়ে গিয়েছি!

এখন প্রশ্ন হল এই কাজটা কোডে করব কিভাবে? লুপ চালিয়ে একটা একটা বিট পরীক্ষা করে করা যেতে পারে কিন্তু কাজ O(1) এই করা যায়। idx নম্বর ইনডেক্সের পরের ইনডেক্সটা হবে:

idx=idx – (idx & -idx)

এটা কিভাবে কাজ করে বোঝার জন‍্য খাতাকলমে বিয়োগটা করে দেখ আমার ব‍্যাখ‍্যা পড়ার আগেই! আমি একটা উদাহরণ দিচ্ছি, ধরি idx=১০ বা বাইনারিতে ১০১০। তাহলে 2′s complement পদ্ধতিতে -idx বা -১০ হবে ০১০১+১=০১১০। (যারা ভুলে গেছ তাদের জন‍্য, 2′s কমপ্লিমেন্ট বের করতে বিটগুলোকে উল্টে দিয়ে ১ যোগ করতে হয়।) এবার যদি বিটওয়াইজ AND করি তাহলে পাবো:

1010  
0110  
——-  
0010

সবথেকে ডানের ১ থেকে বাকি অংশ পেয়ে গেলাম! এবার বিয়োগ করে দিলেই কাজ শেষ। এটা কেন কাজ করছে? আমরা বাইনারি সংখ‍্যাটাকে ৩টা অংশে ভাগ করতে পারি x1z। 1 হলো সবথেকে ডানের ১, x হলো তার আগের যেকোন বিট, z হলো ১ এর পরের থাকা o বিট(x বা z এ শূণ‍্যটা, একটা বা একাধিক বিট থাকতে পারে। এখন বিটগুলো উল্টেদিলে পাবো: x’0z’। x বা z অংশের বিটগুলোকে উল্টে দিয়ে পেয়েছি x’ আর z’, তারমানে z’ হলো কিছু ১ বিট। 2′s কমপ্লিমেন্ট বের করতে ১ যোগ করলে পাবো: x’1z। এখন আগের সংখ‍্যা x1z আর x’1z কে AND করলে থাকে 1z। x এর ভাগটা পুরোটা শূণ‍্য হয়ে যাবে তাই সেটা বাদ। তাহলে আমরা ডানের ১থেকে বাকি অংশ পেয়ে যাবো!

কোডটা তাহলে খুব সহজ:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | int query(int idx)  {         int sum=0;         while(idx>0)         {               sum+=tree[idx];               idx -= idx & (-idx);         }         return sum;    } |

আপডেট অপারেশন কিভাবে করব? ১০১০ বা ১০ এর পরের কোন ইনডেক্সটা ১০ কে কভার করবে? এবার আমরা সবথেকে ডানের ১ কে বামপাশে শিফট করে দিব। ১০১০ এর সাথে ১০ যোগ করলে পাই ১১০০ বা ১২ যেটা ১০কে পুরোটা কভার করে। ঠিক সেরকম ১২ বা ১১০০ এর সাথে ১০০ যোগ করলে পাই ১০০০০ যেটা ১২কে পুরোটা কভার করে। এটা কিভাবে কাজ করছে সেটা চিন্তা করে বের করে ফেল!

আগের মত করেই শুধু বিয়োগের জায়গায় যোগ করলেই সবথেকে ডানের ১টা বামে সরে যাবে। তারমানে যে সংখ‍্যাটা যোগ করছি, নতুন ইনডেক্স থেকে কুয়েরির সময় তারথেকেও বড় একটা সংখ‍্যা বিয়োগ করা হবে তাই নতুন ইনডেক্সটা আগেরটাকে কভার করতে বাধ‍্য! কোডটা হবে এরকম:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | int update(int idx, int x, int n) //n is the size of the array, x is the number to add  {         while(idx<=n)         {               tree[idx]+=x;               idx += idx & (-idx);         }         return sum;    } |

এটাই হলো বাইনারি ইনডেক্সড ট্রি। এখন আমরা সহজেই কোন একটা ইনডেক্স আপডেট করতে পারব এবং 1 থেকে কোন ইনডেক্স পর্যন্ত যোগফল বের করতে পারব।

**কমপ্লেক্সিটি:**

আপডেট এবং কুয়েরি সব ক্ষেত্রেই ২ এর পাওয়ার যোগ বা বিয়োগ করছি, প্রথমে ছোট পাওয়ার থেকে শুরু করে বড় পাওয়ার বিয়োগ করছি। তাই টাইম কমপ্লেক্সিটি হবে O(logn)। শুধুমাত্র Tree অ‍্যারেটাতেই সব অপারেশন হচ্ছে, স্পেস কমপ্লেক্সিটি O(n)।

লক্ষ কর যে এই প্রবলেমটা সেগমেন্ট ট্রি দিয়ে করা গেলেও এখানে মেমরি লাগছে অনেক কম, কোডও খুব ছোট। তবে [সেগমেন্ট ট্রি](http://www.shafaetsplanet.com/planetcoding/?p=1557)তে বিভিন্ন সব রেঞ্জের ভিতর আপডেট করা যায়, এখানে করা যায় ১ নম্বর ইনডেক্স সহ রেঞ্জে।

**প্রবলেম ১:**কুয়েরি করার সময় i থেকে j ইনডেক্সের মধ‍্যের সাবঅ‍্যারের যোগফল বের করতে বললে কি করবে?

**প্রবলেম ২:**কুয়েরিতে যদি বলা হয় অ‍্যারের সর্বনিম্ন কোন ইনডেক্সে গেলে যোগফল X এর বেশি পাবো তাহলে ইনডেক্সটা কিভাবে বের করবে? হিন্টস: O(logn\*logn)।

অনলাইন জাজ থেকে কিছু প্রবলেম:

[Curious Robin Hood](http://lightoj.com/volume_showproblem.php?problem=1112)[Inversion Count](http://www.spoj.com/problems/INVCNT/)  
[Nice Day](http://www.spoj.com/problems/NICEDAY/)

২ডি বাইনারি ইনডেক্সড ট্রি এবং আরো কিছু প্রবলেম নিয়ে ভবিষ‍্যতে লেখার ইচ্ছা আছে। আজকে এই পর্যন্তই। হ‍্যাপি কোডিং!

[Print Friendly](http://www.printfriendly.com/print?url=http://www.shafaetsplanet.com/planetcoding/?p=1961)

ফেসবুকে মন্তব্য

1 comments

Powered by [Facebook Comments](http://peadig.com/wordpress-plugins/facebook-comments/)

ট্যাগ: [ডাটা স্ট্রাকচার](http://www.shafaetsplanet.com/planetcoding/?tag=%e0%a6%a1%e0%a6%be%e0%a6%9f%e0%a6%be-%e0%a6%b8%e0%a7%8d%e0%a6%9f%e0%a7%8d%e0%a6%b0%e0%a6%be%e0%a6%95%e0%a6%9a%e0%a6%be%e0%a6%b0)• [বাইনারি ইনডেক্সড ট্রি](http://www.shafaetsplanet.com/planetcoding/?tag=%e0%a6%ac%e0%a6%be%e0%a6%87%e0%a6%a8%e0%a6%be%e0%a6%b0%e0%a6%bf-%e0%a6%87%e0%a6%a8%e0%a6%a1%e0%a7%87%e0%a6%95%e0%a7%8d%e0%a6%b8%e0%a6%a1-%e0%a6%9f%e0%a7%8d%e0%a6%b0%e0%a6%bf)

Abdul Alim

২৬ আগস্ট ২০১৪

ভাইয়া, ১২ নম্বর ইনডেক্স কি (৯,১২) সাবঅ‍্যারেকে কভার করে না??

শাফায়েত

৪ সেপ্টেম্বর ২০১৪

হ‍্যা। ছবিতে মনে হয় দেখে (৮,১২) মনে হচ্ছে, (৯,১২) হবে।

Maruf

২৬ আগস্ট ২০১৪

Update and Query এর while condition টা উল্টো হয়ে গেছে মনে হয়।

:)

শাফায়েত

৪ সেপ্টেম্বর ২০১৪

ঠিক করে দিলাম। অনেক ধন‍্যবাদ।

Md. Nuhu Alam

২ সেপ্টেম্বর ২০১৪

Is index 5 not covered by index 6?

শাফায়েত

৪ সেপ্টেম্বর ২০১৪

It is covered.

রিপন কুমার সাহা

১৯ সেপ্টেম্বর ২০১৪

ভাইয়া, আপনার পোস্টটা আনেক সুন্দর হয়েছে।

picture এর সাইজ আনেক বেশি হওয়াই পেজ লোড হতে আনেক টাইম নিচ্ছে। picture গুলো একটু ছোট করে দিলে আনেক ভাল হত।

মন্তব্য করুন(বাংলা ভাষায় বাংলা ফন্টে বা ইংরেজি ভাষায় ইংরেজি ফন্টে)

নাম(আবশ্যক)

ইমেইল(আবশ্যক)

ওয়েবসাইট

4 + 1 =

(ইংরেজী ফন্টে বাংলা মন্তব্য মুছে ফেলা হতে পারে)