

Contoh Praktikum Algoritma dan Struktur Data



Nama : Agil Deriansyah Hasan
Nim : 4522210125

Dosen Pengajar :

Dra.SRI REZEKI CANDRA NURSARI,M.Kom
Prak. Algoritma dan Struktur Data - I

**S1-Teknik Informatika
Fakultas Teknik
Universitas Pancasila 2023/2024**

cnthprak10-1

```
asdl1.cpp x asdl1-2.cpp x
1 #include <iostream>
2 using namespace std;
3
4 class btreenode{
5     int *kunci, t, n ;
6     bool leaf;
7     btreenode **c;
8
9     public:
10    btreenode(int tt, bool _leaf);
11    void sisiptdkpenuh(int k);
12    void splitanak(int i, btreenode *y);
13    void traverse();
14    btreenode *search(int k);
15    friend class btree;
16 };
17
18 class btree{
19     int t;
20     btreenode *root;
21
22     public :
23     btree (int tt)
24     {root=NULL;t=tt;}
25     void traverse()
26     {if (root != NULL) root -> traverse();}
27
28     btreenode* search(int k)
29     {return (root == NULL)?NULL : root -> search(k);}
30     void sisip(int k);
31 };
32
33 btreenode::btreenode(int t1, bool leaf1){
34     t = t1;
35     leaf = leaf1;
36     kunci = new int[2*t-1];
37     c = new btreenode *[2*t];
38     n=0;
39 }
40
41 void btreenode::traverse(){
42     int i;
43     for (i=0;i<n;i++){
44         if(leaf==false)
45             c[i]->traverse();
46         cout<<" " <<kunci[i];
47     }
48     if(leaf==false)
49         c[i]->traverse();
50 }
51
```

```
asdl1.cpp x asdl1-2.cpp x
50 }
51
52 btreenode *btreenode::search(int k){
53     int i=0;
54     while(i<n && k>kunci[i])
55         i++;
56     if(kunci[i]==k)
57         return this;
58     if(leaf == true)
59         return NULL;
60     return c[i]->search(k);
61 }
62
63 void btree::sisip(int k){
64     if(root == NULL){
65         root = new btreenode(t,true);
66         root->kunci[0]=k;
67         root->n=1;
68     }
69     else{
70         if(root->n==2*t-1){
71             btreenode *s =new btreenode(t,false);
72             s->c[0]=root;
73             s->splitanak(0,root);
74             int i = 0;
75             if(s->kunci[0]<k){
76                 i++;
77                 s->c[i]->sisiptdkpenuh(k);
78                 root=s;
79             }
80             else
81                 root->sisiptdkpenuh(k);
82         }
83     }
84 }
```

```

85 btreenode::sisiptdkpenuh(int k){
86     int i= n-1;
87     if(leaf == true){
88         while(i>=0 && kunci[i]>k){
89             kunci[i+1]=kunci[i];
90             i--;
91         }
92         kunci[i+1]=k;
93         n=n+1;
94     }
95     else{
96         while(i>=0&&kunci[i]>k)
97             i--;
98         if(c[i+1]->n==2*t-1){
99             splitanak(i+1, c[i+1]);
100             if(kunci[i+1]<k)
101                 i++;
102         }
103         c[i+1]->sisiptdkpenuh(k);
104     }}
105
106 void btreenode::splitanak(int i, btreenode *y){
107     btreenode *z = new btreenode(y->t,y->leaf);
108     z->n=t-1;
109     for(int j=0;j<t-1;j++)
110         z->kunci[j]=y->kunci[j+t];
111     if(y->leaf==false){
112         for(int j=0;j<t;j++)
113             z->c[j]=y->c[j+t];
114     }
115     y->n=t-1;
116     for(int j=n;j>=i+1;j--)
117         c[j+1]=c[j];
118     c[i+1]=z;
119     for(int j=n-1;j>=i;j--)
120         kunci[j+1]=kunci[j];
121     kunci[i]=y->kunci[t-1];
122     n=n+1;
123 }
124

```

```

120     kunci[j+1]=kunci[j];
121     kunci[i]=y->kunci[t-1];
122     n=n+1;
123 }
124
125 int main(){
126     btree t(5);
127     t.sisip(40);
128     t.sisip(49);
129     t.sisip(7);
130     t.sisip(89);
131     t.sisip(20);
132     t.sisip(66);
133     t.sisip(71);
134     t.sisip(75);
135     t.sisip(31);
136     t.sisip(56);
137     t.sisip(81);
138     cout<<"    Pohon Dengan Menggunakan B-Tree    "<<endl;
139     cout<<"----- m=5 -----"<<endl;
140     cout<<"-----"<<endl;
141     t.traverse();
142     cout<<endl;
143     int k =7;
144     (t.search(k) != NULL)?cout<<"Kunci yang dicari "<< k << " = Ditemukan" : cout<<"Kunci yang dicari "<<k<<"= Tidak Ditemukan";
145     cout<<endl;
146     cout<<endl;
147     k=15;
148     (t.search(k) != NULL)?cout<<"Kunci yang dicari "<< k << " = Ditemukan" : cout<<"Kunci yang dicari "<<k<<"= Tidak Ditemukan";
149     return 0;
150 }
151

```

```
F:\>g++ asd11.cpp -o 1
F:\>1
Pohon Dengan Menggunakan B-Tree
-----
m=5
-----
7 20 31 40 49 56 66 71 75 81 89
Kunci yang dicari 7= Ditemukan
Kunciyang dicari 15= Tidak Ditemukan
```

Pseudocode :

Kamus/Deklarasi variabel class btreenode

```
*kunci, t, n, tt, i, k = int
leaf = bool
btree = friend class
```

Kamus/Deklarasi variabel class btree

```
t, tt, k = int
```

Algoritma/Deskripsi class btree

```
public :
{root=NULL;t=tt;}
{if (root != NULL) root -> traverse();}
btreenode* search(int k)
{return (root == NULL)?NULL : root -> search(k);}
```

Kamus/Deklarasi Variabel fungsi btreenode

```
t1 = int
leaf1 = bool
```

Algoritma/Deskripsi fungsi btreenode(int t1, bool leaf1)

```
t = t1
leaf = leaf1
kunci = new int[2*t-1]
c = new btreenode *[2*t]
n=0
```

Kamus/Deklarasi Variabel fungsi traverse

```
i = int
```

Algoritma/Deskripsi fungsi traverse

```
for (i=0;i<n;i++)
    if(leaf==false)
        c[i]->traverse()
        kunci[i]
endfor
if(leaf==false)
    c[i]->traverse()
```

Kamus/Deklarasi variabel fungsi search

```
k, i = int
```

Algoritma/Deskripsi fungsi search (k)

```
while(i<n && k>kunci[i])
    i++
if(kunci[i]==k)
    return this
if(leaf == true)
    return NULL
return c[i]->search(k)
```

Kamus/Deklarasi Variabel fungsi sisip

```
k = int
```

Algoritma/Deskripsi fungsi sisip

```
if(root == NULL)
    root = new btreenode(t,true)
    root->kunci[0]=k
    root->n=1
else
    if(root->n==2*t-1)
        btreenode *s =new btreenode(t,false)
        s->c[0]=root
        s->splitanak(0,root)
        i = 0
        if(s->kunci[0]<k)
            i++
            s->c[i]->sisiptdkpenuh(k)
            root=s
        else
            root->sisiptdkpenuh(k)
    endif
endif
```

Kamus/Deklarasi Variabel fungsi sisisptdkpenuh
i, k = int

Algoritma/Deskripsi fungsi sisisptdkpenuh(k)

```
i=n-1
if(leaf == true)
    while(i>=0 && kunci[i]>k)
        kunci[i+1]=kunci[i]
        i--
    endwhile
    kunci[i+1]=k
    n=n+1
else
    while(i>=0&&kunci[i]>k)
        i--
    if(c[i+1]->n==2*t-1)
        splitanak(i+1, c[i+1])
        if(kunci[i+1]<k)
            i++
        endif
    endif
    c[i+1]->sisisptdkpenuh(k)
```

```
        k=7
(t.search(k) != NULL)?; print k ; print k
        k = 15
(t.search(k) != NULL)?; print k ; print k
return 0
```

Kamus/Deklarasi Variabel fungsi splitanak
i,j=int

Algoritma/Deskripsi fungsi splitanak(i,btreenode *y)

```
btreenode *z = new btreenode(y->t,y->leaf);
z->n=t-1;
for(int j=0;j<t-1;j++)
    z->kunci[j]=y->kunci[j+t]
if(y->leaf==false)
    for(int j=0;j<t;j++)
        z->c[j]=y->c[j+t]
endif
y->n=t-1
for(int j=n;j>=i+1;j--)
    c[j+1]=c[j]
    c[i+1]=z
for(int j=n-1;j>=i;j--)
    kunci[j+1]=kunci[j]
    kunci[i]=y->kunci[t-1]
n=n+1
```

Kamus/Deklarasi Variabel fungsi utama
k = int

Algoritma/Deskripsi fungsi utama

```
btree t(5);
t.sisip(40);
t.sisip(49);
t.sisip(7);
t.sisip(89);
t.sisip(20);
t.sisip(66);
t.sisip(71);
t.sisip(75);
t.sisip(31);
t.sisip(56);
t.sisip(81);
t.traverse()
```

Algoritma :

```
1.  Membuat fungsi btreenode(t1,leaf1)
2.  t = t1
3.  leaf = leaf1
4.  kunci = new int[2*t-1]
5.  c = new btreenode *[2*t]
6.  n=0
7.  membuat fungsi traverse
8.  Selama (i=0) maka kerjakan baris 9 s.d 12
9.  Jika (leaf==false)
10. c[i] -> traverse
11. Mencetak Nilai kunci[i]
12. i++
13. Jika (leaf==false)
14. c[i]->traverse()
15. Membuat fungsi search(k)
16. i=0
17. Selama (i<n && k>kunci[i])
18. i++
19. Jika (kunci[i]==k)
20. return this
21. Jika (leaf == true)
22. return NULL
23. return c[i]->search(k)
24. Membuat fungsi sisip (k)
25. Jika (root == NULL) maka kerjakan baris
    26 s.d 28 kalau tidak 29 s.d 38
26. root = new btreenode(t,true)
27. root->kunci[0]=k
28. root->n=1

29. Jika (root->n==2*t-1) maka kerjakan baris
    30 s.d 37 kalau tidak baris 38
30. btreenode *s =new btreenode(t,false)
31. s->c[0]=root
32. s->splitanak(0,root)
33. i = 0
34. Jika (s->kunci[0]<k)
35. i++
36. s->c[i]->sisiptdkpenuh(k)
37.root=s
38. root->sisiptdkpenuh(k)
39. Membuat fungsi sisiptdkpenuh (k)
40. i=n-1
41. Jika(leaf==true) maka kerjakan baris 42
    s.d 46 kalau tidak 47 s.d
42. Selama (i>=0 && kunci[i]>k) 43 s.d 44
43. kunci[i+1]=kunci[i];
44. i--
45. kunci[i+1]=k
46. n=n+1
47. Selama (i>=0&&kunci[i]>k)
48. i--
49. Jika (c[i+1]->n==2*t-1) maka kerjakan
    baris 50 s.d 52
50. splitanak(i+1, c[i+1])
51. Jika(kunci[i+1]<k)
52. i++
53. c[i+1]->sisiptdkpenuh(k)
```

```
54. Membuat fungsi splitanak(i,*y)
55. btreenode *z = new btreenode(y->t,y->leaf)
56. z->n=t-1
57. Selama ( j=0)
58. z->kunci[j]=y->kunci[j+t]
59. j++
60. Jika (y->leaf==false) maka kerjakan baris 61 s.d 62
61. Selama ( j=0)
62. z->c[j]=y->c[j+t]
63. y->n=t-1
64. Selama (j=n)
65. c[j+1]=c[j]
66. c[i+1]=z
67. j-
68. Selama (j=n-1)
69. kunci[j+1]=kunci[j]
70. kunci[i]=y->kunci[t-1]
71. n=n+1
72. j-
73. Membuat fungsi utama
74. btree t(5)
75. t.sisip(40)
76. t.sisip(49)
77. t.sisip(7)
78. t.sisip(89)
79. t.sisip(20)
80. t.sisip(66)
81. t.sisip(71)
82. t.sisip(75)
83. t.sisip(31)
84. t.sisip(56)
85. t.sisip(81)
86. Memanggil fungsi t.traverse()
87. k=7
88. (t.search(k) != NULL)?
89. Mencetak nilai k
90. k=15
91. (t.search(k) != NULL)?
92. Mencetak nilai k
93. Selesai
```

cnthprak10-2

```
asd11.cpp x asd11-2.cpp x
1 #include <iostream>
2 using namespace std;
3
4 struct canbtree {
5     int *d;
6     canbtree **cananakpointer;
7     bool l;
8     int n;
9 } *r = NULL, *np = NULL, *x = NULL;
10
11 canbtree* init() {
12     int i;
13     np = new canbtree;
14     np->d = new int[6];
15     np->cananakpointer = new canbtree*[7];
16     np->l = true;
17     np->n = 0;
18     for (i = 0; i < 7; i++) {
19         np->cananakpointer[i] = NULL;
20     }
21     return np;
22 }
23
24 void pohonb(canbtree *p) {
25     if (!p) return;
26     int i;
27     for (i = 0; i < p->n; i++) {
28         if (!p->l) {
29             pohonb(p->cananakpointer[i]);
30         }
31         cout << " " << p->d[i];
32     }
33     if (!p->l) {
34         pohonb(p->cananakpointer[i]);
35     }
36     cout << endl;
37 }
38
39 void urut(int *p, int n) {
40     int i, j, t;
41     for (i = 0; i < n - 1; i++) {
42         for (j = i + 1; j < n; j++) {
43             if (p[i] > p[j]) {
44                 t = p[i];
45                 p[i] = p[j];
46                 p[j] = t;
47             }
48         }
49     }
50 }
51
```

```
asd11.cpp x asd11-2.cpp x
50 F:\asd11.cpp
51
52 int pecahanak(canbtree *x, int i) {
53     int j, mid;
54     canbtree *np3;
55     np3 = init();
56     np3->l = true;
57
58     if (i == -1) {
59         mid = x->d[2];
60         x->d[2] = 0;
61         x->n--;
62     }
63     return mid;
64 }
65
66 void sisip(int a) {
67     int i, t;
68     x = r;
69     if (x == NULL) {
70         r = init();
71         x = r;
72     } else {
73         if (x->l == true && x->n == 6) {
74             t = pecahanak(x, -1);
75             x = r;
76             for (i = 0; i < (x->n); i++) {
77                 if ((a > x->d[i]) && (a < x->d[i + 1])) {
78                     i++;
79                     break;
80                 } else if (a < x->d[0]) {
81                     break;
82                 } else {
83                     continue;
84                 }
85             }
86             x = x->cananakpointer[i];
87         } else {
88             while (x->l == false) {
89                 for (i = 0; i < (x->n); i++) {
90                     if ((a > x->d[i]) && (a < x->d[i + 1])) {
91                         i++;
92                         break;
93                     } else if (a < x->d[0]) {
94                         break;
95                     } else {
96                         continue;
97                     }
98                 }
99             }
100         }
101     }
102 }
```



```
asd11.cpp x asd11-2.cpp x
85     }
86     x = x->cananakpointer[i];
87 } else {
88     while (x->l == false) {
89         for (i = 0; i < (x->n); i++) {
90             if ((a > x->d[i] && (a < x->d[i + 1])) {
91                 i++;
92                 break;
93             } else if (a < x->d[0]) {
94                 break;
95             } else {
96                 continue;
97             }
98         }
99         if ((x->cananakpointer[i])->n == 6) {
100             t = pecahanak(x, i);
101             x->d[x->n] = t;
102             x->n++;
103             urut(x->d, x->n);
104         } else {
105             x = x->cananakpointer[i];
106         }
107     }
108 }
109 }
110 x->d[x->n] = a;
111 urut(x->d, ++x->n);
112 }
113
114 int main() {
115     int i, n, t;
116     cout << "Masukkan Jumlah Elemen yang akan diinput = ";
117     cin >> n;
118     cout << endl;
119     for (i = 0; i < n; i++) {
120         cout << "Masukkan Isi Elemen = ";
121         cin >> t;
122         sisip(t);
123     }
124     cout << endl;
125     cout << "Hasil Pengurutan Menggunakan Btree" << endl;
126     cout << "~~~~~" << endl;
127     pohonb(r);
128 }
129 }
```

```
Command Prompt
Microsoft Windows [Version 10.0.22631.3593]
(c) Microsoft Corporation. All rights reserved.

C:\Users\agild>F:

F:\>g++ asd11-2.cpp -o 1

F:\>1
Masukkan Jumlah Elemen yang akan diinput = 5

Masukkan Isi Elemen = 23
Masukkan Isi Elemen = 1
Masukkan Isi Elemen = 2
Masukkan Isi Elemen = 35
Masukkan Isi Elemen = 321

Hasil Pengurutan Menggunakan Btree
~~~~~
1 2 23 35 321

F:\>|
```

Algoritma :

1. Membuat fungsi canbtree* init
2. Deklarasi struktur (struct { *d,l,n,**cananakpointer)
3. Mendefinisikan *r = NULL, *np = NULL, *x = NULL
4. np = new canbtree
5. np->d = new int[6]
6. np->cananakpointer = new canbtree*[7]
7. np->l = true
8. np->n = 0
9. Selama (i = 0) maka kerjakan baris 10 s.d 11
10. np ->cananakpointer[i]=NULL
11. i++
12. return np
13. Membuat fungsi pohonb(canbtree *p)
14. Selama (i=0) maka kerjakan baris 15 s.d 20
15. Jika (p->==false) maka kerjakan baris 16
16. pohonB(p->cananakpointer[i])
17. Mencetak p -> d[i]
18. Jika (p->l==false) maka kerjakan baris 19
19. pohonB(p->cananakpointer[i])
20. i++
21. Membuat fungsi urut(*p, n)
22. Selama (i=0) maka kerjakan baris 23 s.d 29
23. selama (j=i) maka kerjakan baris 24 s.d 28
24. Jika (p[i] > p[j]) maka kerjakan baris 25 s.d 27
25. t = p[i]
26. p[i] = p[j]
27. p[j] = t
28. j++
29. i++
30. Membuat fungsi pecahanak(canbtree *x, i)
31. canbtree *np3
32. np3=init
33. np3->l=true
34. Jika (i==1) maka kerjakan baris 35 s.d 37
35. mid = x->d[2]
36. x->d[2] = 0
37. x->n--

38. return mid
39. Membuat fungsi sisip (a)
40. x=r
41. Jika (x==NULL) maka kerjakan baris 42 s.d 43 kalau tidak 44 s.d 70
42. r=init
43. x=r
44. Jika(x->l == true && x->n == 6) maka kerjakan baris 45 s.d 55 kalau tidak baris 56 s.d 70
45. t = pecahanak(x, -1)
46. x = r
47. selama (i=0) maka kerjakan baris 48 s.d 54
48. Jika ((a > x->d[i]) && (a < x->d[i + 1])) maka kerjakan baris 49 s.d 50 kalau tidak 51 s.d 53
49. i++
50. break
51. Jika (a < x->d[0]) maka kerjakan baris 52 kalau tidak baris 53
52. break
53. continue
54. i++
55. x = x->cananakpointer[i]
56. Selama (x->1==false) maka kerjakan baris 57 s.d 6
57. Selama (i=0) maka kerjakan baris 58 s.d 64
58. Jika ((a > x->d[i]) && (a < x->d[i + 1])) maka kerjakan baris 59 s.d 60 kalau tidak 61 s.d 63
59. i++
60. break
61. jika(a < x->d[0]) maka kerjakan baris 62 kalau tidak 63
62. break
63. continue
64. i++
65. Jika ((x->cananakpointer[i])->n == 6) maka kerjakan baris 66 s.d 69 kalau tidak 70
66. t = pecahanak(x, i)
67. x->d[x->n] = t
68. x->n++
69. urut(x->d, x->n)
70. x = x->cananakpointer[i]

71. x->d[d->n]=a
72. urut(x->d,x->n)
73. x->n++
74. Membuat fungsi utama
75. Menginput/Memasukkan nilai n
76. Selama (i=0) maka kerjakan baris 77 s.d 79
77. Menginput/Memasukkan nilai t
78. Memanggil fungsi sisip(t)
79. i++
80. Memanggil fungsi pohonb(r)

Pseudocode

Kamus/Deklarasi Variabel fungsi init
i = int

Algoritma/Deklarasi fungsi init
np = new canbtree
np->d = new int[6]
np->cananakpointer = new canbtree*[7]
np->l = true
np->n = 0
for (i = 0; i < 7; i++)
 np->cananakpointer[i] = NULL
endfor
return np

Kamus/Deklarasi Variabel fungsi pohonb
*p = canbtree
i = int

Algoritma/Deklarasi fungsi pohonb(canbtree *p)
if (!p) return
 for (i = 0; i < p->n; i++)
 if (!p->l)
 pohonb(p->cananakpointer[i])
 endfor
 print p->d[i]
endif
if (!p->l)
 pohonb(p->cananakpointer[i])
endif

Kamus/Deklarasi variabel fungsi urut
*p, n, i, j, t = int

Algoritma/Deskripsi fungsi urut (*p, n)
for (i = 0; i < n - 1; i++)
 for (j = i + 1; j < n; j++)
 if (p[i] > p[j])
 t = p[i]
 p[i] = p[j]
 p[j] = t
 endif
 endfor
endfor

Kamus/Deklarasi Variabel fungsi pecahanak
*x = canbtree
i, j, mid = int

Algoritma/Deskripsi fungsi pecahanak (canbtree *x, i)
canbtree *np3
np3 = init()
np3->l = true
 if (i == -1)
 mid = x->d[2]
 x->d[2] = 0
 x->n--
 endif
return mid

Kamus/Deklarasi Variabel fungsi sisip
a, i, t = int

```
else
  while (x->l == false)
    for (i = 0; i < (x->n); i++)
      if ((a > x->d[i]) && (a < x->d[i + 1]))
        i++
        break
      else if (a < x->d[0])
        break
      else
        continue
    endfor
    if ((x->cananakpointer[i])->n
    == 6)
      t = pecahanak(x, i)
      x->d[x->n] = t
      x->n++
      urut(x->d, x->n)
      else
      x = x->cananakpointer[i]
      x->d[x->n] = a
      urut(x->d, ++x->n)
      endif
    endwhile
  endif
endif
```

Algoritma/Deskripsi fungsi sisip (a)

```
x = r
if (x == NULL)
  r = init()
  x = r
else
  if (x->l == true && x->n == 6)
    t = pecahanak(x, -1)
    x = r
    for (i = 0; i < (x->n); i++)
      if ((a > x->d[i]) && (a < x->d[i + 1]))
        i++
        break
      else if (a < x->d[0])
        break
      else
        continue
      endif
    endfor
    x = x->cananakpointer[i]
```

Kamus/Deklarasi Variabel fungsi utama
i,n,t = int

Algoritma/Deskripsi fungsi utama

```
input n
for (i = 0; i < n; i++)
  input t
  sisip(t)
endfor
pohonb(r)
```