

Contents

| | | |
|-----|---------------|---|
| 1 | Basic | 1 |
| 1.1 | ascii | 1 |
| 1.2 | limits | 1 |
| 1.3 | priorityqueue | 1 |
| 1.4 | graph | 1 |
| 2 | Section2 | 2 |
| 2.1 | thm | 2 |

1 Basic

1.1 ascii

| int | char | int | char | int | char |
|-----|------|-----|------|-----|------|
| 32 | | 64 | @ | 96 | ` |
| 33 | ! | 65 | A | 97 | a |
| 34 | " | 66 | B | 98 | b |
| 35 | # | 67 | C | 99 | c |
| 36 | \$ | 68 | D | 100 | d |
| 37 | % | 69 | E | 101 | e |
| 38 | & | 70 | F | 102 | f |
| 39 | ' | 71 | G | 103 | g |
| 40 | (| 72 | H | 104 | h |
| 41 |) | 73 | I | 105 | i |
| 42 | * | 74 | J | 106 | j |
| 43 | + | 75 | K | 107 | k |
| 44 | , | 76 | L | 108 | l |
| 45 | - | 77 | M | 109 | m |
| 46 | . | 78 | N | 110 | n |
| 47 | / | 79 | O | 111 | o |
| 48 | 0 | 80 | P | 112 | p |
| 49 | 1 | 81 | Q | 113 | q |
| 50 | 2 | 82 | R | 114 | r |
| 51 | 3 | 83 | S | 115 | s |
| 52 | 4 | 84 | T | 116 | t |
| 53 | 5 | 85 | U | 117 | u |
| 54 | 6 | 86 | V | 118 | v |
| 55 | 7 | 87 | W | 119 | w |
| 56 | 8 | 88 | X | 120 | x |
| 57 | 9 | 89 | Y | 121 | y |
| 58 | : | 90 | Z | 122 | z |
| 59 | ; | 91 | [| 123 | { |
| 60 | < | 92 | \ | 124 | |
| 61 | = | 93 |] | 125 | } |
| 62 | > | 94 | ^ | 126 | ~ |
| 63 | ? | 95 | _ | | |

1.2 limits

| [Type] | [size] | [range] |
|--------------------|--------|---|
| char | 1 | 127 to -128 |
| signed char | 1 | 127 to -128 |
| unsigned char | 1 | 0 to 255 |
| short | 2 | 32767 to -32768 |
| int | 4 | 2147483647 to -2147483648 |
| unsigned int | 4 | 0 to 4294967295 |
| long | 4 | 2147483647 to -2147483648 |
| unsigned long | 4 | 0 to 18446744073709551615 |
| long long | 8 | 9223372036854775807 to -9223372036854775808 |
| double | 8 | 1.79769e+308 to 2.22507e-308 |
| long double | 16 | 1.18973e+4932 to 3.3621e-4932 |
| float | 4 | 3.40282e+38 to 1.17549e-38 |
| unsigned long long | 8 | 0 to 18446744073709551615 |
| string | 32 | |

1.3 priorityqueue

```

1 priority_queue<int> pq;
2
3 pq.push(x);
4
5 x = pq.top();
6 pq.pop();           // delete after read
7
8 pq.empty()          //return true
9 pq.size()           //return 0
10
11 priority_queue<T> pq;           //from big to small
12 priority_queue<T, vector<T>, greater<T> > pq;
13                               //from small to big
14 priority_queue<T, vector<T>, cmp> pq; //cmp

```

1.4 graph

```

1
2 #include<bits/stdc++.h>
3 using namespace std;
4
5 class Node {
6 public:
7     int val;
8     vector<Node*> children;
9
10    Node() {}
11
12    Node(int _val) {
13        val = _val;
14    }
15
16    Node(int _val, vector<Node*> _children) {
17        val = _val;
18        children = _children;
19    }
20 };
21
22 struct ListNode {
23     int val;
24     ListNode *next;
25     ListNode() : val(0), next(nullptr) {}
26     ListNode(int x) : val(x), next(nullptr) {}
27     ListNode(int x, ListNode *next) : val(x),
28         next(next) {}
29 };
30
31 struct TreeNode {
32     int val;
33     TreeNode *left;
34     TreeNode *right;
35     TreeNode() : val(0), left(nullptr),
36         right(nullptr) {}
37     TreeNode(int x) : val(x), left(nullptr),
38         right(nullptr) {}
39     TreeNode(int x, TreeNode *left, TreeNode *right)
40         : val(x), left(left), right(right) {}
41 };
42
43 class ListProblem {
44 public:
45     void solve() {
46         return;
47     }
48
49     ListNode* buildList(int idx) {
50         if(idx == nums.size()) return NULL;
51         ListNode *current=new
52             ListNode(nums[idx++],current->next);
53         return current;
54     }
55
56     void deleteList(ListNode* root) {
57         if(root == NULL) return;
58         deleteList(root->next);
59     }

```

```

55     delete root;
56     return;
57 }
58 };
59
60 class TreeProblem {
61     int null = INT_MIN;
62     vector<int> nums = {}, result;
63 public:
64     void solve() {
65
66         return;
67     }
68
69     TreeNode* buildBinaryTreeUsingDFS(int left, int
70         right) {
71         if((left > right) || (nums[(left+right)/2] ==
72             null)) return NULL;
73         int mid = (left+right)/2;
74         TreeNode* current = new TreeNode(
75             nums[mid],
76             buildBinaryTreeUsingDFS(left, mid-1),
77             buildBinaryTreeUsingDFS(mid+1, right));
78         return current;
79     }
80
81     TreeNode* buildBinaryTreeUsingBFS() {
82         int idx = 0;
83         TreeNode* root = new TreeNode(nums[idx++]);
84         queue<TreeNode*> q;
85         q.push(root);
86         while(idx < nums.size()) {
87             if(nums[idx] != null) {
88                 TreeNode* left = new
89                     TreeNode(nums[idx]);
90                 q.front()->left = left;
91                 q.push(left);
92             }
93             idx++;
94             if((idx < nums.size()) && (nums[idx] !=
95                 null)) {
96                 TreeNode* right = new
97                     TreeNode(nums[idx]);
98                 q.front()->right = right;
99                 q.push(right);
100             }
101             idx++;
102             q.pop();
103         }
104         return root;
105     }
106
107     Node* buildNAryTree() {
108         int idx = 2;
109         Node *root = new Node(nums.front());
110         queue<Node*> q;
111         q.push(root);
112         while(idx < nums.size()) {
113             while((idx < nums.size()) && (nums[idx]
114                 != null)) {
115                 Node *current = new Node(nums[idx++]);
116                 q.front()->children.push_back(current);
117                 q.push(current);
118             }
119             idx++;
120             q.pop();
121         }
122         return root;
123     }
124
125     void deleteBinaryTree(TreeNode* root) {
126         if(root->left != NULL)
127             deleteBinaryTree(root->left);
128         if(root->right != NULL)
129             deleteBinaryTree(root->right);
130         delete root;
131         return;
132     }

```

```

124     }
125
126     void deleteNAryTree(Node* root) {
127         if(root == NULL) return;
128         for(int i=0; i<root->children.size(); i++) {
129             deleteNAryTree(root->children[i]);
130             delete root->children[i];
131         }
132         delete root;
133         return;
134     }
135
136     void inorderTraversal(TreeNode* root) {
137         if(root == NULL) return;
138         inorderTraversal(root->left);
139         cout<<root->val<<' ';
140         inorderTraversal(root->right);
141         return;
142     }
143 };
144
145 int main() {
146
147     return 0;
148 }

```

2 Section2

2.1 thm

- 中文測試

$$\sum_{i=1}^n i^2 = \frac{n(n+1)(2n+1)}{6}$$