Contents

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

```cpp
priority_queue <int> pq;

pq.push(x);

x = pq.top();
pq.pop(); // delete after read

pq.empty() //return true
pq.size() //return 0

priority_queue<T> pq; //from big to small
priority_queue<T, vector<T>, greater<T> > pq;   //from
    small to big
priority_queue<T, vector<T>, cmp> pq; //cmp
```

1.4 graph

```cpp
#include<bits/stdc++.h>
using namespace std;

class Node {
public:
    int val;
    vector<Node*> children;

    Node() {}

    Node(int _val) {
        val = _val;
    }

    Node(int _val, vector<Node*> _children) {
        val = _val;
        children = _children;
    }
};

struct ListNode {
    int val;
    ListNode *next;
    ListNode() : val(0), next(nullptr) {}
    ListNode(int x) : val(x), next(nullptr) {}
    ListNode(int x, ListNode *next) : val(x),
        next(next) {}
};

struct TreeNode {
    int val;
    TreeNode *left;
    TreeNode *right;
    TreeNode() : val(0), left(nullptr),
        right(nullptr) {}
    TreeNode(int x) : val(x), left(nullptr),
        right(nullptr) {}
    TreeNode(int x, TreeNode *left, TreeNode *right)
        : val(x), left(left), right(right) {}
};

class ListProblem {
    vector<int> nums={};
public:
    void solve() {
        return;
    }

    ListNode* buildList(int idx) {
        if(idx == nums.size()) return NULL;
        ListNode *current=new
            ListNode(nums[idx++],current->next);
        return current;
    }

    void deleteList(ListNode* root) {
        if(root == NULL) return;
        deleteList(root->next);
```

```cpp
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
            right) {
70          if((left > right) || (nums[(left+right)/2] ==
                null)) return NULL;
71          int mid = (left+right)/2;
72          TreeNode* current = new TreeNode(
73              nums[mid],
74              buildBinaryTreeUsingDFS(left,mid-1),
75              buildBinaryTreeUsingDFS(mid+1,right));
76          return current;
77      }
78
79      TreeNode* buildBinaryTreeUsingBFS() {
80          int idx = 0;
81          TreeNode* root = new TreeNode(nums[idx++]);
82          queue<TreeNode*> q;
83          q.push(root);
84          while(idx < nums.size()) {
85              if(nums[idx] != null) {
86                  TreeNode* left = new
                        TreeNode(nums[idx]);
87                  q.front()->left = left;
88                  q.push(left);
89              }
90              idx++;
91              if((idx < nums.size()) && (nums[idx] !=
                    null)) {
92                  TreeNode* right = new
                        TreeNode(nums[idx]);
93                  q.front()->right = right;
94                  q.push(right);
95              }
96              idx++;
97              q.pop();
98          }
99          return root;
100     }
101
102     Node* buildNAryTree() {
103         int idx = 2;
104         Node *root = new Node(nums.front());
105         queue<Node*> q;
106         q.push(root);
107         while(idx < nums.size()) {
108             while((idx < nums.size()) && (nums[idx]
                    != null)) {
109                 Node *current = new Node(nums[idx++]);
110                 q.front()->children.push_back(current);
111                 q.push(current);
112             }
113             idx++;
114             q.pop();
115         }
116         return root;
117     }
118
119     void deleteBinaryTree(TreeNode* root) {
120         if(root->left != NULL)
                deleteBinaryTree(root->left);
121         if(root->right != NULL)
                deleteBinaryTree(root->right);
122         delete root;
123         return;
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

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