Day Cover

6230362521 Patcharapong K. (I got covid krub)

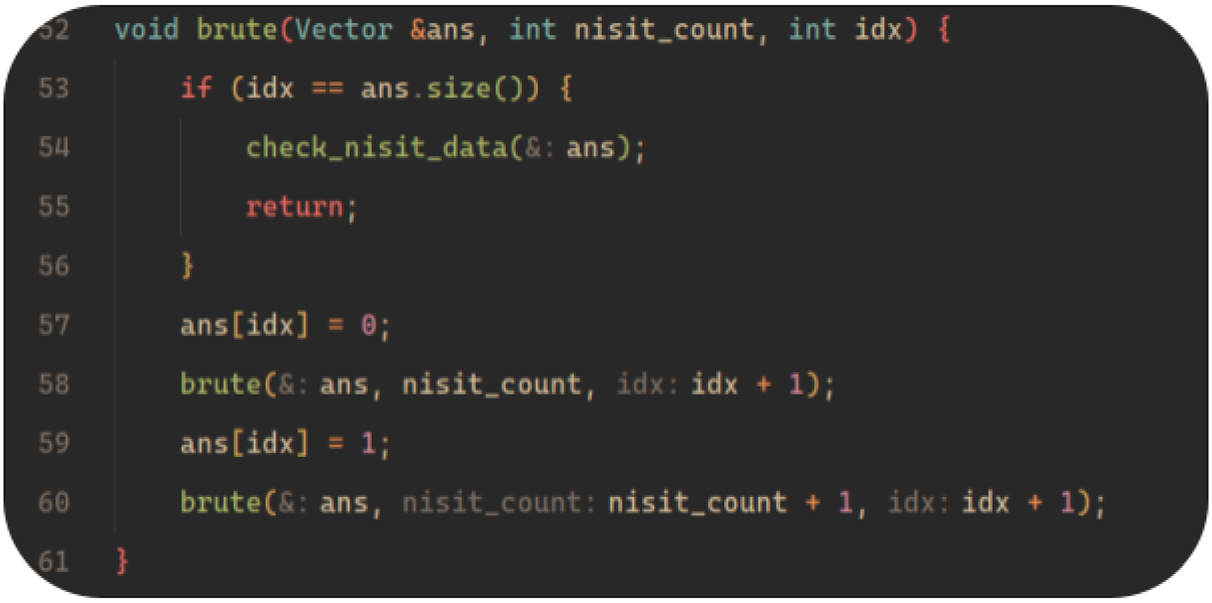
Initial Code

```
#include <iostream>
#include <map>
#include <vector>

using namespace std;
typedef vector<int> Vector;
typedef map<int, Vector> Map;

int n;  // days to cover
int m;  // nisit count

Map nisit_map;  // info of free days per nisit

Vector days_to_cov;
int least_ppl = 99999999;

void print_vec(Vector &v) {
   for (const int &x : v)
       cout << x << " ";
   cout << endl;
}
void reset_vec(Vector &v) {
   for (int i = 0; i < v.size(); ++i) {
       v[i] = 0;
   }
}

bool check_all_cov() {
   for (int i = 0; i < n; ++i) {
       if (days_to_cov[i] == 0)
           return false;
   }
   return true;
}
int count_ones(Vector &v) {
   int ones = 0;
   for (auto x : v)
       if (x == 1)
           ones++;
   return ones;
}

void check_nisit_data(Vector &ans) {
```

```cpp
    int nisit_count = 0;
    reset_vec(days_to_cov);
    for (int i = 0; i < ans.size(); ++i) {
        // nisit is chosen
        if (ans[i] == 1) {
            nisit_count++;
            for (int x : nisit_map[i]) {
                days_to_cov[x - 1] += 1;
            }
        }
    }
    bool covered_all_days = check_all_cov();
    if (covered_all_days) {
        int ones_count = count_ones(ans);
        if (ones_count < least_ppl) {
            least_ppl = ones_count;
        }
    }
}

void brute(Vector &ans, int n) {
    if (n == ans.size()) {
        check_nisit_data(ans);
        return;
    }
    ans[n] = 0;
    brute(ans, n + 1);
    ans[n] = 1;
    brute(ans, n + 1);
}

void get_inputs() {
    cin >> n >> m;
    days_to_cov.resize(n);
    int day;
    for (int i = 0; i < m; ++i) {
        cin >> day;
        int free_day;
        for (int k = 0; k < day; ++k) {
            cin >> free_day;
            nisit_map[i].push_back(free_day);
        }
    }
}

int main() {
    get_inputs();
    Vector ans(m);
    brute(ans, 0);
    cout << least_ppl << endl;
    return 0;
}
```

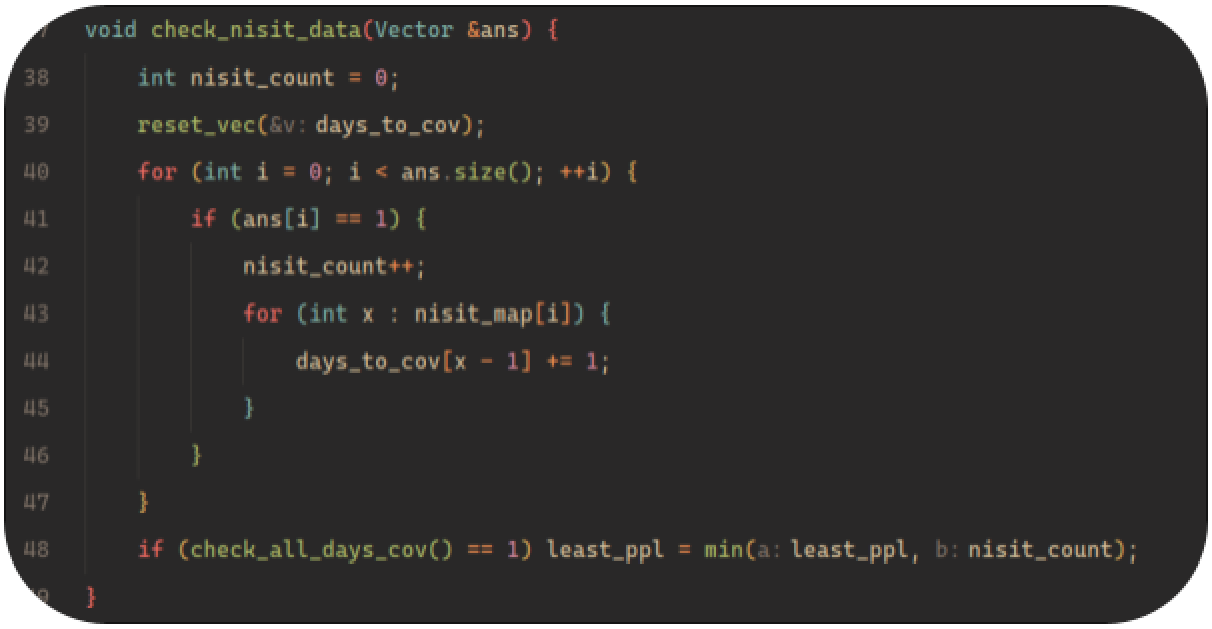
Potential Problem(s)

The code above led to 65/100 with the 35 points left being Time limit exceeded (T). The problem is with the nature of exhausting the entire answer set:



```
void brute(Vector &ans, int nisit_count, int idx) {
    if (idx == ans.size()) {
        check_nisit_data(&: ans);
        return;
    }
    ans[idx] = 0;
    brute(&: ans, nisit_count, idx: idx + 1);
    ans[idx] = 1;
    brute(&: ans, nisit_count: nisit_count + 1, idx: idx + 1);
}
```

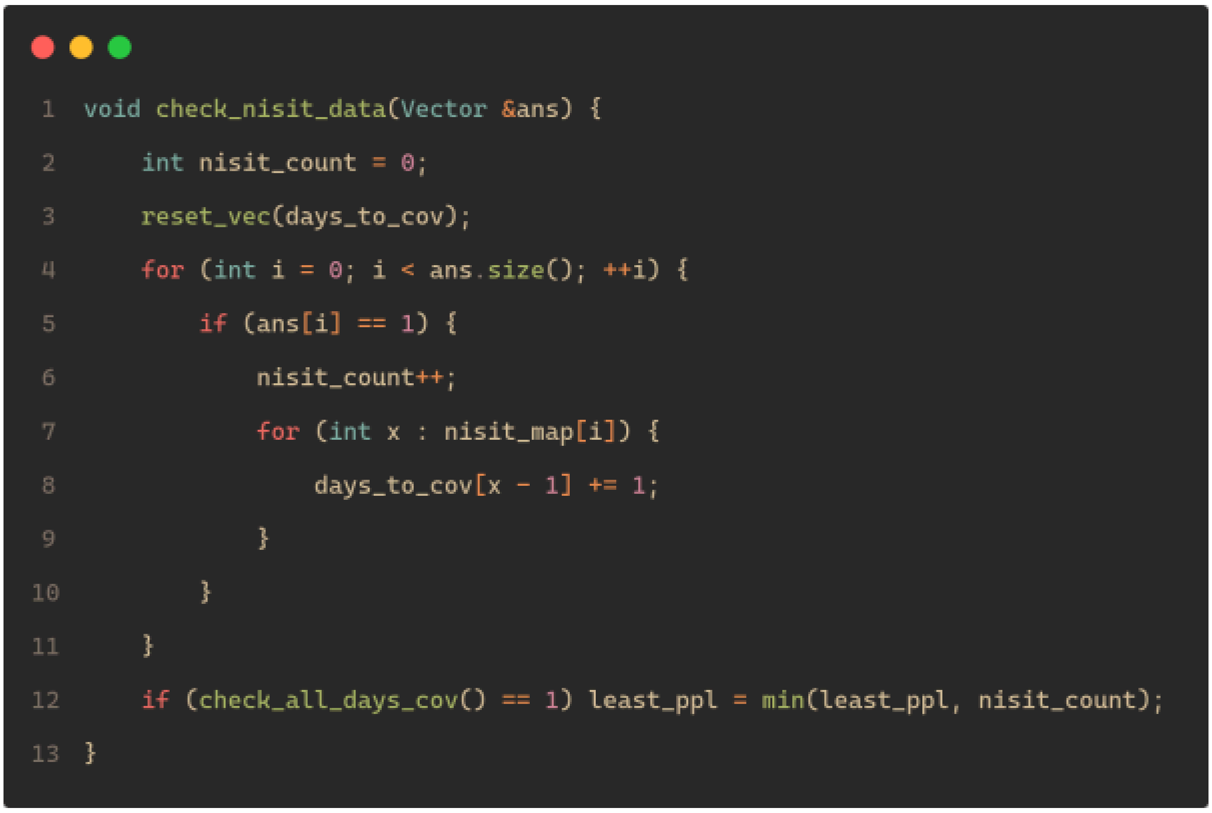
As suggested by Aj.Nattee, we should prune the branch more. I tried adding if the current trial leads to more nisit_count, we early exit.



```
void check_nisit_data(Vector &ans) {
    int nisit_count = 0;
    reset_vec(&v: days_to_cov);
    for (int i = 0; i < ans.size(); ++i) {
        if (ans[i] == 1) {
            nisit_count++;
            for (int x : nisit_map[i]) {
                days_to_cov[x - 1] += 1;
            }
        }
    }
    if (check_all_days_cov() == 1) least_ppl = min(a: least_ppl, b: nisit_count);
}
```

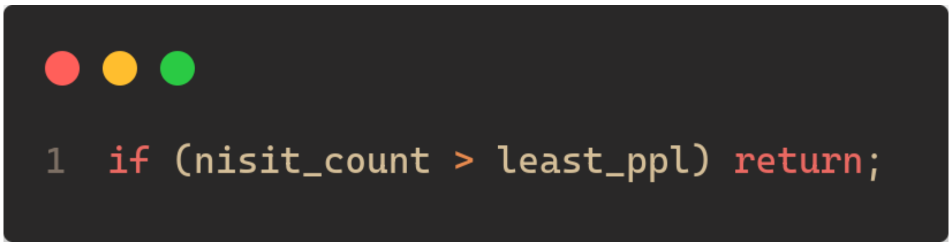
The grading system still gives me 65/100, which means this alone is not enough.

The Check Data Function



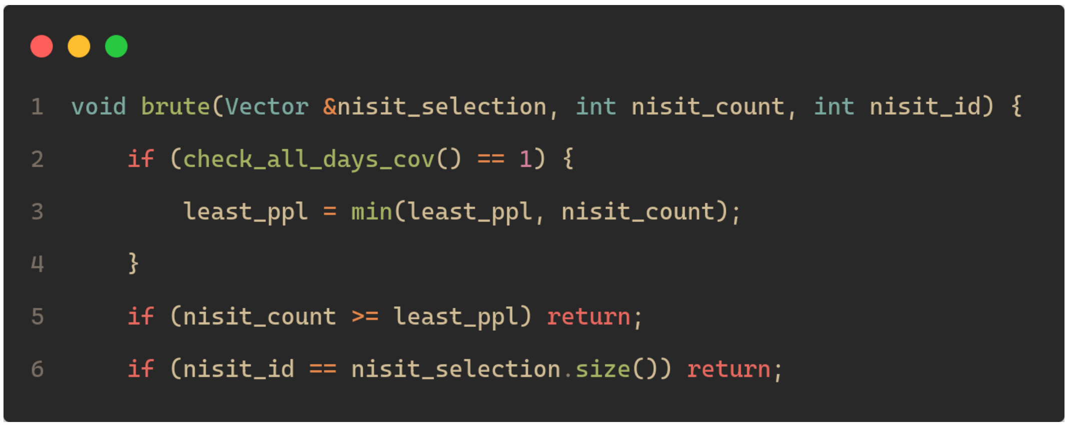
```
void check_nisit_data(Vector &ans) {
    int nisit_count = 0;
    reset_vec(days_to_cov);
    for (int i = 0; i < ans.size(); ++i) {
        if (ans[i] == 1) {
            nisit_count++;
            for (int x : nisit_map[i]) {
                days_to_cov[x - 1] += 1;
            }
        }
    }
    if (check_all_days_cov() == 1) least_ppl = min(least_ppl, nisit_count);
}
```

Since there is nothing left to be done in the “brute force” function. The problem should lie on the check data function. Here we can see that it took O(n) for resetting a vector, O(n^2) for checking the day coverage and another O(n) for checking if all day is covered.



```
if (nisit_count > least_ppl) return;
```

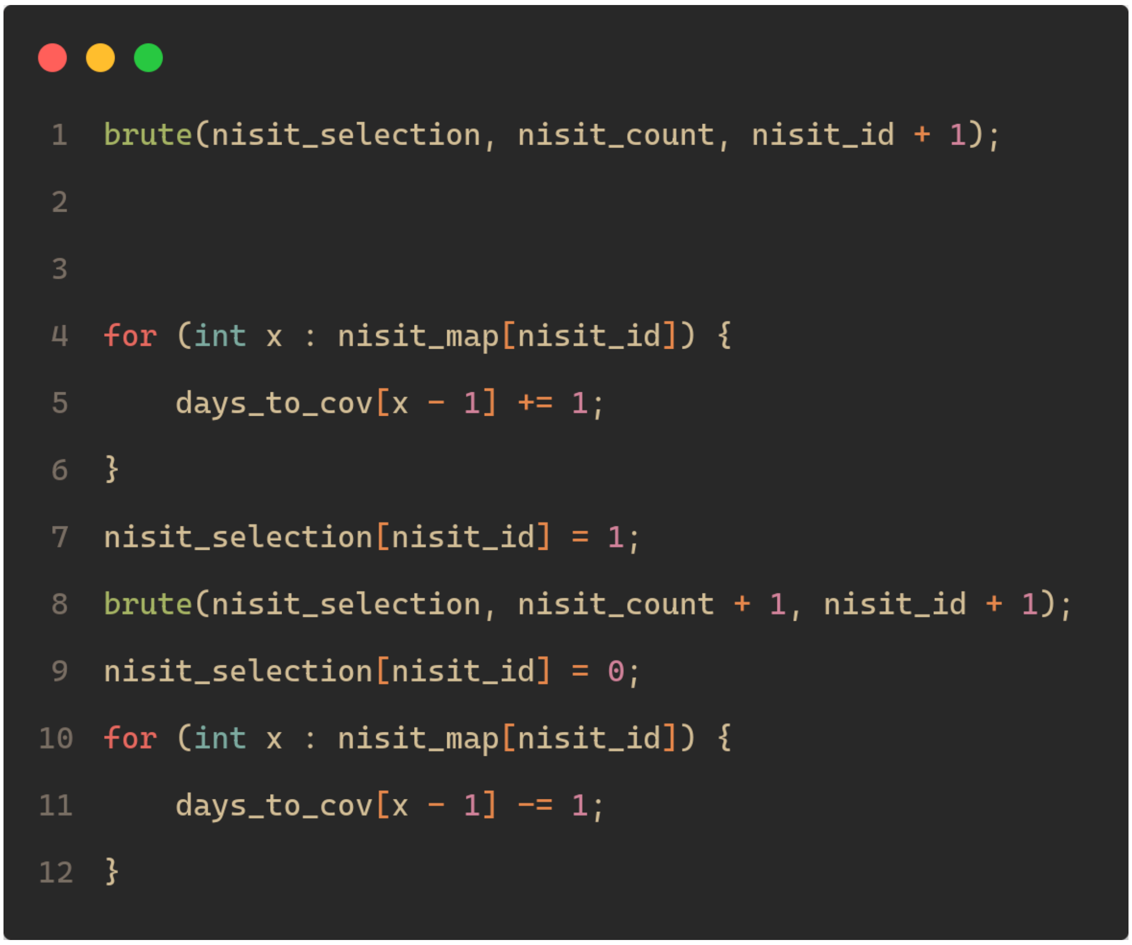
This line should be changed to >= since there is no point repeating the minimum nisit count again. Now the function becomes as follows:



```cpp
void brute(Vector &nisit_selection, int nisit_count, int nisit_id) {
    if (check_all_days_cov() == 1) {
        least_ppl = min(least_ppl, nisit_count);
    }
    if (nisit_count >= least_ppl) return;
    if (nisit_id == nisit_selection.size()) return;
```

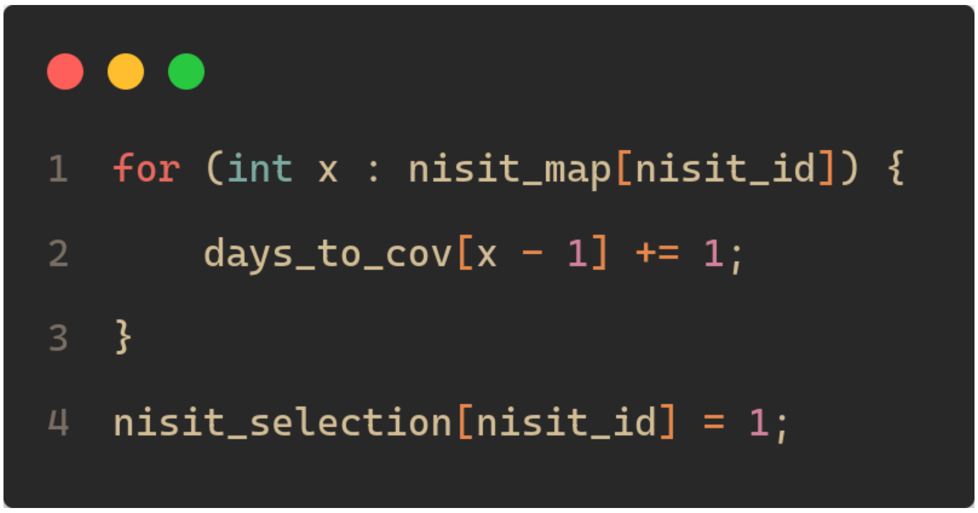
Note that variable names were changed to avoid confusion.

Here, every round we check if the days are already covered. If the current nisit count is greater than or equal to the global least count, we just return and traverse other branches.



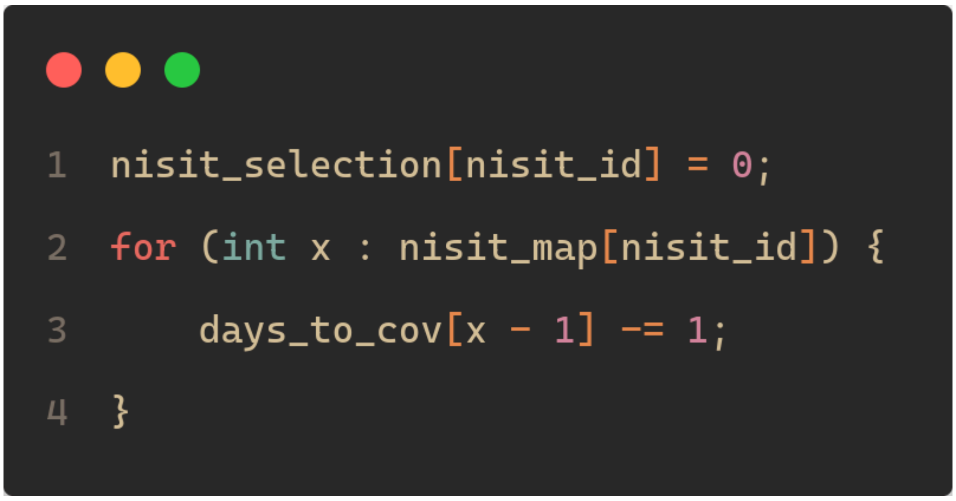
```cpp
brute(nisit_selection, nisit_count, nisit_id + 1);


for (int x : nisit_map[nisit_id]) {
    days_to_cov[x - 1] += 1;
}
nisit_selection[nisit_id] = 1;
brute(nisit_selection, nisit_count + 1, nisit_id + 1);
nisit_selection[nisit_id] = 0;
for (int x : nisit_map[nisit_id]) {
    days_to_cov[x - 1] -= 1;
}
```

For the recursive part, I changed from re-evaluating the entire "nisit_selection" vector to just increase/decrease the day coverages per nisit.



```
for (int x : nisit_map[nisit_id]) {
    days_to_cov[x - 1] += 1;
}
nisit_selection[nisit_id] = 1;
```

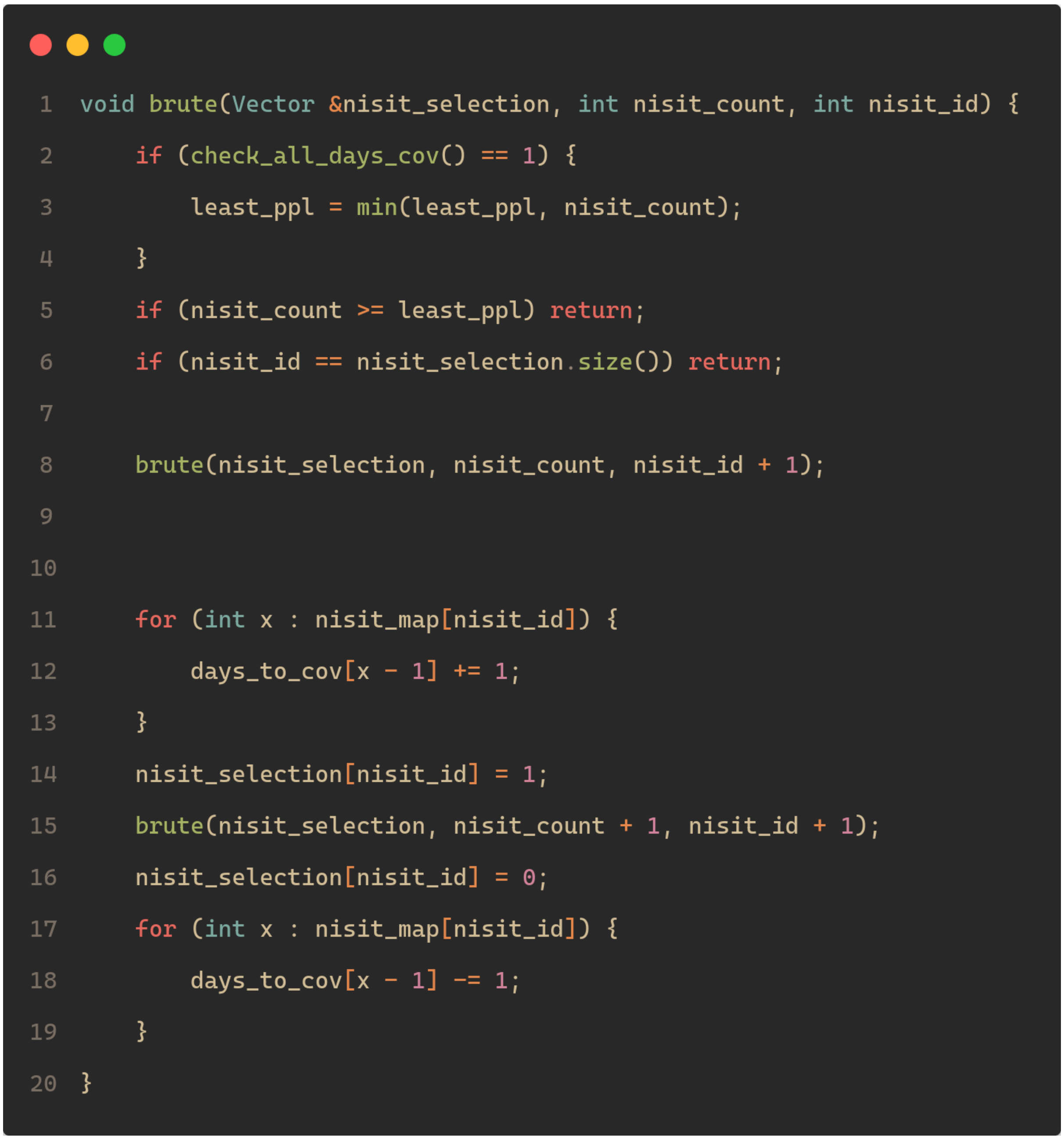
Here, if the nisit is selected, the days_to_cov (days coverage) vector should update accordingly.



```
nisit_selection[nisit_id] = 0;
for (int x : nisit_map[nisit_id]) {
    days_to_cov[x - 1] -= 1;
}
```

When we are done selecting that nisit, we remove them out and decrease the day coverage of them as well.

This way ensures that we don't recalculate the day coverage every time. We simply calculate the coverage at this point in time (selected/not-selected) and remove out when not using.



```
void brute(Vector &nisit_selection, int nisit_count, int nisit_id) {
    if (check_all_days_cov() == 1) {
        least_ppl = min(least_ppl, nisit_count);
    }
    if (nisit_count >= least_ppl) return;
    if (nisit_id == nisit_selection.size()) return;

    brute(nisit_selection, nisit_count, nisit_id + 1);


    for (int x : nisit_map[nisit_id]) {
        days_to_cov[x - 1] += 1;
    }
    nisit_selection[nisit_id] = 1;
    brute(nisit_selection, nisit_count + 1, nisit_id + 1);
    nisit_selection[nisit_id] = 0;
    for (int x : nisit_map[nisit_id]) {
        days_to_cov[x - 1] -= 1;
    }
}
```

Full new function

Full 100/100 Code

```cpp
#include <iostream>
#include <map>
#include <vector>

using namespace std;
typedef vector<int> Vector;
typedef map<int, Vector> Map;

int n; // days to cover
int m; // nisit count

Map nisit_map; // info of free days per nisit

Vector days_to_cov;
int least_ppl = 99999999;

int check_all_days_cov() {
 for (auto x : days_to_cov) {
   if (x == 0)
     return 0;
 }
 return 1;
}

void brute(Vector &nisit_selection, int nisit_count, int nisit_id) {
   if (check_all_days_cov() == 1) {
       least_ppl = min(least_ppl, nisit_count);
   }
   if (nisit_count >= least_ppl) return;
   if (nisit_id == nisit_selection.size()) return;

   brute(nisit_selection, nisit_count, nisit_id + 1);


   for (int x : nisit_map[nisit_id]) {
       days_to_cov[x - 1] += 1;
   }
   nisit_selection[nisit_id] = 1;
   brute(nisit_selection, nisit_count + 1, nisit_id + 1);
   nisit_selection[nisit_id] = 0;
   for (int x : nisit_map[nisit_id]) {
       days_to_cov[x - 1] -= 1;
   }
}

void get_inputs() {
 cin >> n >> m;
 days_to_cov.resize(n, 0); // Initialize with zeros
 for (int i = 0; i < m; ++i) {
   int day;
   cin >> day;
   for (int k = 0; k < day; ++k) {
     int free_day;
     cin >> free_day;
     nisit_map[i].push_back(free_day);
   }
 }
}
```

```
int main() {
 get_inputs();
 Vector nisit_selection(m);
 brute(nisit_selection, 0, 0);
 cout << least_ppl << endl;
 return 0;
}
```