Підготовка до сертифікаційного тесту Samsung

Для підготовки до тесту ми радимо вам прочитати матеріал нижче та розв'язати наведені задачі. Вам знадобляться знання з динамічного програмування, а також розуміння діаметру графа.

Загальна інформація

Сертифікаційний тест – це 1 задача з програмування, яку необхідно розв'язати за 4 години на C, C++ або Java. Після реєстрації у online-системі ваш програмний код проходить автоматичне тестування (на вхід система дає тестові значення і перевіряє результати ваших обчислень). Якщо усі тести пройдено успішно, задача вважається розв'язаною.

У програмному коді можна використовувати лише базові конструкції мови: знання високорівневих пакетів не потрібне. Дозволено використання лише наступних бібліотек:
C : <stdio.h>, <malloc.h>
C++ : <stdio.h>, <malloc.h>, <iostream.h>
Java : java.util.Scanner
Обмеження на використання стандартних бібліотек зумовлено необхідністю перевірки базових навичок та розуміння базових алгоритмів у кандидата.

Також не рекомендуємо використовувати динамічне виділення пам'яті. Краще користуватись статичними масивами максимального розміру згідно з обмеженнями умови задачі.

В якості середовища розробки використовуються
C, C++ : Visual Studio
Java : Eclipse
Тому при підготовці до тесту ми радимо користуватись саме цими середовищами.

При розв'язанні задачі ми радимо перші 40-60 хвилин виділити на читання та розуміння задачі, та побудови математичної моделі. Наступні 2 години можна витратити на написання коду, щоб залишити годину або півтори на пошук дефектів та корекцію коду. Радимо написати декілька власних тестів.

Зверніть увагу! Кількість спроб для проходження тесту обмежена – ви зможете спробувати здати його тільки 3 рази. Тож краще підготуватись перед тим, як намагатись здати тест.

Сертифікаційна система.
Для того, щоб ви могли легше адаптуватись до сертифікаційної системи, ми радимо вам зареєструватись та вирішити принаймні задачу на сайті codeground.org. В розділі https://www.codeground.org/practice знайдіть через пошук, наприклад, просту задачу "Hello" та вирішіть її. Текст задачі дасть вам уявлення по можливі помилки в

англійському тексті в умовах задачі, а вікно для завантаження коду в систему на codeground.org схоже на сертифікаційну систему Samsung.

Тренування.

Для тренування ми пропонуємо вам вирішити наступні задачі:
(динамічне програмування) "Отрезки":
https://github.com/algosolver01/algorithms/tree/master/Отрезки
(динамічне програмування) "Сломанный калькулятор":
https://github.com/algosolver01/algorithms/tree/master/Сломанный%20калькулятор
(динамічне програмування) "Башня":
https://github.com/algosolver01/algorithms/tree/master/Башня
(діаметр графу) "Матожидание диаметра дерева":
http://codeforces.com/problemset/problem/804/D
За складністю вони приблизно відповідають задачам сертифікаційного тексту.

Динамічне програмування.
Більшість задач, які раніше зустрічались на тесті, були на динамічне програмування. Якщо ви не часто з ним зустрічаєтесь, ми радимо вам подивитись, наприклад, цю статтю, щоб згадати основи динамічного програмування:
https://habrahabr.ru/post/113108/

Більше теоретичних матеріалів за потреби можна знайти тут:
https://habrahabr.ru/post/191498/
http://stu.sernam.ru/book_sop.php?id=14
http://informatics.mccme.ru/course/view.php?id=9

Діаметр графу.
Ще одна тема, яка може зустрітись на тесті – це діаметр графу.
Диаметром графа называют число d, равное расстоянию между наиболее удаленными друг от друга вершинами графа.
В общем случае, вычисляется расстояние между всеми парами вершин и находится максимум (например, алгоритмом Флойда — Уоршелла за O(V^3)).
Дерево - связный граф без циклов. Равносильное определение: связный граф, в котором количество ребер на 1 меньше количества вершин.
Алгоритм поиска диаметра дерева (неориентированного графа) за O(V)

- Выбираем произвольную вершину
- Обходом графа, находим наиболее удаленную от нее вершину А.
- Находим наиболее удаленную от А вершину Б.

Утверждается, что расстояние от А до Б – диаметр графа.

Приклад рішення (C++)

```
struct Edge {
    int to, w;
    Edge *next;
};
int edges;
```

```
Edge* vertex[kMaxN];

int dst[kMaxN + 5]; //rule of thumb: add extra cells
int stack[kMaxN + 5];
int dfs(int f) {
    const int INF = kMaxN * kMaxW + 5;
    for(int i = 0; i < n; ++i) dst[i] = INF;
    dst[f] = 0;
    int top = 0;
    stack[top++] = f;
    int ans = f;
    while (top) {
        int cur = stack[--top];
        if (dst[ans] < dst[cur]) ans = cur;
        Edge *e = vertex[cur];
        while (e) {
            int to = e->to;
            int w = dst[cur] + e->w;
            if (dst[to] > w) {
                dst[to] = w;
                stack[top++] = to;
            }
            e = e->next;
        }
    }
    return ans;
}
int diam(int f) {
    return dst[dfs(dfs(f))];
}
```

Підтримка.
З будь-якими питанням по задачам ви можете звернутись до експерта Самсунг – Андрія Заболотного, a.zabolotnyi@samsung.com.
Не соромтесь звертатись, він буде радий вам допомогти!

Щасти вам на тесті!