Desarrollo e implementación de un programa de trabajo para el semillero de programación

Por:
Ana Echavarría Uribe

Tutor:
Juan Francisco Cardona Mc'Cormick

Universidad EAFIT

31 de mayo de 2013

Contenido

¿Qué es el Semillero de Programación?

- Grupo de la Universidad donde se preparan los alumnos para participar en las maratones de programación realizadas por ACIS/REDIS y por la ACM-ICPC.
- En tres ocasiones, alumnos del Semillero han clasificado a la Maratón Mundial de Programación ACM-ICPC.
- El semillero ha estado a cargo de alumnos destacados en las maratones de programación.
- No tiene un plan de trabajo implementado.

Problema

- Este semestre no había quién dirigiera el Semillero.
- Vi la necesidad y la ventaja de crear un programa de trabajo que permitiera que los estudiantes aprendieran los algoritmos más útiles para las maratones.

¿Cómo desarrollar el plan de trabajo?

Para desarrollar el plan de trabajo fue necesario resolver tres problemas

- ¿Qué metodología usar para las sesiones?
- ¿Qué temas enseñar en cada sesión?
- ¿Cómo aplicar los algoritmos enseñados a problemas tipo maratón de programación?

Curso Competitive Programming

- Curso creado en 2008 en la Universidad Nacional de Singapur que busca fortalecer las habilidades de programación de sus estudiantes destacados para prepararlos para las competencias universitarias de programación de la ICPC
- Enfocado a estudiantes de tercer año con buenas habilidades de programación
- Material abierto al público en `https://sites.google.com/site/stevenhalim/home/material`

Curso Competitive Programming

- ~~Exámenes~~ $\rightarrow$ Competencias
- Enseña algoritmos haciendo énfasis en su implementación eficiente y aplicaciones en lugar de enfocarse en las pruebas de corrección y de complejidad.
- Al igual que el Semillero, busca crear un espacio en el que los estudiantes puedan prepararse para las competencias de programación.
- En varias ocasiones, estudiantes de este curso han llegado a competir en la Maratón Mundial de Programación ACM-ICPC.

Estructura de cada sesión

Cada sesión consta de tres partes

1. Discusión y solución de los problemas propuestos como tarea en la sesión anterior.
2. Exposición del nuevo tema a trabajar, mostrando los algoritmos, los elementos matemáticos relacionados y sus implementaciones en el lenguaje C++.
3. Presentación breve de los problemas propuestos como ejercicio para la siguiente sesión.

¿Qué temas enseñar?

- Seleccionar un conjunto de temas a enseñar que fueran importantes para las maratones.
- Escoger apropiadamente el orden de enseñanza para facilitar la comprensión de cada tema.
- Buscar problemas de cada tema para hacer las competencias.

¿Qué temas enseñar?

- Una fracción de los temas del curso Competitive Programming.
- Algunos de los temas del curso de verano de maratones de programación 2012 de la UNICAMP.
- Temas de libros de preparación para las maratones de programación.
- La selección se hizo de acuerdo a la experiencia en competencias pasadas.

Temas enseñados

Semana	Temas
1	Introducción a las maratón y jueces de programación
2	Arreglos de C++, Vectores de C++ y Grafos
3	Representación de grafos en C++
4	Pila y Cola, BFS y DFS
5	Problemas de BFS y DFS
6	Map, Set, Heap, Ordenamiento topológico Componentes fuertemente conexas
7	Algoritmo de Dijkstra
8	Algoritmo de Bellman-Ford
9	Programación dinámica: Problemas clásicos
10	Algoritmo de Floyd-Warshall
11	Árbol de mínima expansión
12	Algoritmo de Knuth-Morris-Pratt
13	Algoritmo de máximo flujo
14	Solución de problemas de la IV Maratón de Programación UTP
15	Algoritmos de teoría de números

¿Cómo desarrollar las competencias?

- Crear competencias semanales en los que se apliquen los temas de cada sesión.
- Utilizar los archivos de programación de los jueces: UVa, Codeforces y Spoj.
- Seleccionar los problemas de acuerdo a los problemas propuestos en diferentes libros de programación competitiva y otros problemas que se consideren de utilidad.
- Resolver los problemas seleccionados.

Documentación

- Se crearon diapositivas, competencias, soluciones a problemas y un manual con los algoritmos trabajados como material de apoyo para el grupo.
- El contenido se actualiza semana a semana en el repositorio `https://github.com/anaechavarria/SemilleroProgramacion`
- Este contenido se compartió también con los directores de las maratones de programación de la Universidad Tecnológica de Pereira y la Universidad Pontificia Bolivariana para que lo compartieran con sus alumnos.

SemilleroProgramacion /

100 commits

Fixing typos

anaechavarria authored 3 hours ago — latest commit fa406d1364

Diapositivas	3 days ago	Adding partial results to final report [anaechavarria]
Manual	3 hours ago	Fixing typos [anaechavarria]
Trabajo	3 hours ago	Fixing typos [anaechavarria]
.DS_Store	3 days ago	Adding partial results to final report [anaechavarria]
.gitignore	4 days ago	Changing .gitignore [anaechavarria]
README.md	3 days ago	Create README.md [anaechavarria]

SemilleroProgramacion / Diapositivas / **History**

Adding partial results to final report

anaechavarria authored 3 days ago latest commit 7ef4a348c8

..		
1. Arreglos, Vectores, Grafos	3 months ago	Adding PDF after changing arrays for vetors [anaechavarria]
10. Árbol de Mínima Expansión	6 days ago	Adding Prim's algorithm [anaechavarria]
11. Algoritmo de Knuth-Morris-Pratt	6 days ago	Adding string algorithms to notebook [anaechavarria]
12. Algoritmo de Máximo Flujo	6 days ago	Adding maxflow to notebook [anaechavarria]
13. Algoritmos de Teoría de Números	6 days ago	Adding number theoretic algorithms to notebook [anaechavarria]
2. Representación de grafos, getline y strin...	3 months ago	Adding slides for graph representation, getline and stringstream [anaechavarria]
3. Pila, Cola, DFS y BFS	3 months ago	Adding slides for topological sort and scc [anaechavarria]
4. Problemas de DFS y BFS	a month ago	Initial commit of maxflow algorithm [anaechavarria]
5. Ordenamiento Topológico, Componentes...	3 months ago	Adding heap operations [anaechavarria]
6. Algoritmo de Dijkstra	11 days ago	Fixing mistakes on slides [anaechavarria]
7. Algoritmo de Bellman-Ford	2 months ago	Adding Bellman-Ford slides [anaechavarria]
8. Programación Dinámica	6 days ago	Adding LIS and LCS to notebook [anaechavarria]
9. Algoritmo de Floyd-Warshall	a month ago	Fixing typo on minimax formula [anaechavarria]
.DS_Store	3 days ago	Adding partial results to final report [anaechavarria]

Manual de algoritmos del semillero de programación EAFIT

Ana Echavarría

27 de mayo de 2013

Índice

1. Plantilla

```
using namespace std;
#include <algorithm>
#include <iostream>
#include <iterator>
#include <numeric>
#include <sstream>
#include <fstream>
#include <cassert>
#include <climits>
#include <cstdlib>
#include <cstring>
#include <string>
#include <cstdio>
#include <vector>
#include <cmath>
#include <queue>
#include <stack>
#include <list>
#include <map>
#include <set>

// Template para recorrer contenedores usando iteradores
```

3.1. Divisores de un número

Imprime los divisores de un número (cuidado que no lo hace en orden). Complejidad: $O(\sqrt{n})$ donde n es el número.

```
void divisors(int n){
    int i;
    for (i = 1; i * i < n; ++i){
        if (n % i == 0) printf("%d\n%d\n", i, n/i);
    }
    // Si existe, imprimir su raiz cuadrada una sola vez
    if (i * i == n) printf("%d\n", i);
}
```

3.5.1. Propiedades de la operación módulo

- $(a \bmod n) \bmod n = a \bmod n$
- $(a + b) \bmod n = ((a \bmod n) + (b \bmod n)) \bmod n$
- $(a \cdot b) \bmod n = ((a \bmod n) \cdot (b \bmod n)) \bmod n$
- $\left(\frac{a}{b}\right) \bmod n \neq \left(\frac{a \bmod n}{b \bmod n}\right) \bmod n$

Preguntas