Problemas & Jueces

agosto, 2017

Técnicas para resolver problemas

Categorías

En general los problemas a resolver pueden clasificarse de acuerdo a la técnica necesaria para resolverlos. Las categorías son:

- Estructuras de datos avanzadas
- Algoritmos sobre strings
- Geometría computacional
- Problemas de carácter matemático
- Backtracking
- Divide y Conquer
- Programación dinámica
- Greedy algorithms
- Teoría de grafos
- Ad-Hoc

Teoría de Grafos

Dentro de lo que es la teoría de grafos, podemos encontrarnos con los siguientes tipos de problemas:

- Recorridos, conectitud y orden topológico
  - BFS
  - DFS
- Caminos mínimos
  - Dijkstra
  - Bellman-Ford
- Puntos de articulación
- Matching y flujo
  - Ford-Fulkerson
  - Edmonds-Karp
- Arbol generador mínimo
  - Kruskal
  - Prim
- Ejes puente
- Grafos eulerianos

Técnicas para resolver problemas

Utilidad en la vida laboral

- Facilidad en la resolución de problemas complejos.
- Optimización en la era de la eficiencia.
- Habilidad para analizar un problema.

Estructuras avanzadas

Entre las estructuras avanzadas a utilizar en la resolución de los problemas encontramos:

- Tries
- AVL
- Heap
- RMQ
- Árbol de intervalos
- Suffix tree
- Suffix array
- LCP array

Algoritmos sobre strings

Los algoritmos avanzados de strings son:

- Los de matching-perfecto:
  - Rabin-Karp
  - Knuth-Morris-Prat
- Utilización de expresiones regulares
- Parseo de strings

Para realizar muchos de estos algoritmos se tendría que usar una o mas de las estructuras avanzadas indicadas en la diapositiva anterior.

Geometría computacional

Aquellos problemas de geometría computacional hacen uso de los siguientes elementos:

- Manejo de distintos espacios
  - R^n
  - Grillas
  - Superficies
- Trigonometría con punto flotante en aritmética finita
- Geometría proyectiva con enteros
- Operaciones con figuras geométricas elementales

Problemas de carácter matemático

En los problemas matemáticos se van a tener que resolver problemas de:

- Aritmética entera
  - Divisibilidad
  - Primos
  - Factorización
  - MCM y MCD
  - Congruencias
- Aritmética en Polinomios
- Matrices
- Combinatoria
- Probabilidades

Estructuras útiles para investigar

C++ STL

- Vector
- Set
- Map.
- priority_queue(heap)
- Pair.
- Stack
- Queue

Estructuras útiles para investigar

Java

- ArrayList
- HashSet
- HashMap
- TreeSet
- TreeMap.
- Priority-Queue(heap)
- Stack
- Queue

Soluciones a los Problemas

Por lo general existen diversas formas de llegar a la solución planteada por el problema.

Sin embargo se trata de buscar la solución más eficiente en tiempo y en el uso de espacio.

- Diferentes algoritmos de ordenación.
- Encontrar los números primos.

Manejo de Datos

C++

Existen dos formas de manejar la entrada y salida en C++:

- cin - cout
- scanf printf

Manejo de input en C++

Las funciones que nos provee C++ para leer la entrada es:

```
1 istream & operator >>
```

Este operador trabaja sobre un stream de entrada. El stream asociado a la entrada estándar es `std::cin`.
La ventaja de este operador es que lee el stream hasta encontrar el formato indicado por el segundo operando, por lo que usando el mismo operador podemos leer números, strings, caracteres, etc.

```
1 char c ;
2 string s ;
3 int d;
4 std::cin >> s >> c >> d ;
```

En el ejemplo se lee primero un **string**, luego un **caracter** y por último un **entero**.

Manejo de Datos

C++

Manejo de output en C++ Las funciones que nos provee C++ para imprimir la salida es:

```
1  ostream & operator <<
```

Este operador trabaja sobre un stream de salida. El stream asociado a la salida estándar es `std::cout`. Trabaja de forma similar al operador de lectura pero de forma inversa.

```
1 char c = 's' ;
2 strings = "texto de prueba" ;
3 int d = 80;
4 std::cout << s << c << d << std::endl ;
```

Manejo de Datos

C++

Manejo de input/output en C++

Uso de scanf/printf permite un tiempo más eficiente:

Código[GNU]C++	Sample Input	Sample Output

`#include<cstdio>`
`int TC, a, b;`
`scanf("%d", &TC); // nro de casos`
`while (TC − −) { //repite hasta 0`
`scanf("%d%d", &a, &b) ; //calcular la resp`
`printf("%d \n", a + b);`
`}`
	3
	1 2
	5 7
	6 3
		3
		12
		9

Referencia de los tipos de datos: http://www.cplusplus.com/reference/cstdio/scanf/

Manejo de input en Java

Java nos provee una clase **wrapper**: java.util.Scanner

Esta clase provee métodos para extraer distintos tipos de datos de un stream de datos, así como para saber si existen datos en el stream. En nuestro caso usaremos `System.in` como stream que es la entrada estándar.

Para ver cómo se usa entren a: http://docs.oracle.com/javase/1.5.0/docs/api/java/util/Scanner.html

```
1 Scanner in = new Scanner(new BufferedReader(new
      InputStreamReader
      (System.in)));
```

Manejo de output en Java

El manejo de output en java se hace mediante el método `println`, print o format de la clase PrintStream.

El objeto sobre el cual tendremos que llamar los métodos es `System.out` el cual es un `PrintStream` que representa la salida estándar.

La semántica de las funciones es muy simple y puede verse en: http://docs.oracle.com/javase/7/docs/api/java/io/PrintStream.html

```
1 BufferedWriter out = new BufferedWriter(new
      OutputStreamWriter(System.out));
2 out.write("strings");
3 out.flush() // AL FINAL DEL PROGRAMA!
```

Jueces

¿Qué son?

Los jueces son sistemas diseñados para recibir soluciones y corregirlas.

Todos los jueces aplican los mismos criterios de corrección y devuelven el mismo tipo de respuestas.

Para practicar existen varios jueces que están en línea las 24 horas y reciben respuestas para una gran cantidad de problemas.

Jueces

¿Cuáles son?

Entre los jueces más importantes y que son usados por la materia se encuentran:

- UVa Online Judge: Gran diversidad de problemas de competencias locales. http://uva.onlinejudge.org/
- Sphere Online Judge (SPOJ): Juez con una gran cantidad de problemas de todo tipo. http://www.spoj.com/
- COJ: http://coj.uci.cu/index.xhtml
- LiveArchive: Contiene los problemas de las competencias regionales y finales de la ACM-ICPC. http://livearchive.onlinejudge.org/

Jueces

Lenguajes Aceptados

A continuación se indican los lenguajes aceptados por cada juez:

- LiveArchive
  - C
  - C++
  - Java
- UVa Online Judge
  - C
  - C++
  - Java
  - Python
- Sphere Online Judge(SPOJ)
  - C
  - C++
  - Java
  - Python
  - Pascal
  - ... muchos más

Problemas

Características Principales

Todos los problemas en las competencias tienen las siguientes características:

- Están escritos en inglés.
- El planteo de los problemas se hace en forma de "historia", obligando a realizar una abstracción del mismo.
- El enunciado se encuentra divido en: planteo, descripción d la entrada y descripción de la salida.
- No puede asumirse nada acerca de lo que no está explicitado en el enunciado.

Jueces

Posibles Respuestas

El juez retornará alguna de las siguientes respuestas:

- **Accepted**: La solución ingresada es válida.
- **Wrong Answer**: El programa compiló y corrió correctamente pero la salida no es la esperada.
- **Presentation Error**: El programa compiló, corrió y las respuestas son las correctas pero no sigue el formato establecido.
- **Compilation Error**: El código fuente no compila.
- **Runtime Error**: El programa no terminó de forma correcta. (Segmentation Fault, NullPointerException, etc)
- **Time Limit Exceeded**: El programa corrió por más tiempo del permitido. (Loop infinito o la solución no es la óptima)
- **Memory Limit Exceeded**: El programa usó mas memoria de la permitida.

Datos de entrada
En todos los problemas de las competencias los datos de entrada deben ser leídos de la entrada estándar.

El formato en que vienen los datos es explicitado en el enunciado del problema.

En los problemas recientes el formato de los datos ha sido estandarizado y estructurado por lo que es raro que se encuentren con problemas donde tengan que leer toda una línea para después parsearla.

Datos de salida
En todos los problemas de las competencias los datos de salida deben ser impresos a la salida estándar.

El formato en el que hay que imprimir los datos es explicitado en el enunciado del problema.

Prestar mucha atención a los espacios y las líneas en blanco que pide el enunciado. No realizar esto correctamente resulta en **Presentation Error** lo que perjudica innecesariamente el puntaje.

Probando los programas

Datos de prueba

Probar siempre el programa con:

- Datos de entrada dados.
- Puntos límites de entrada.
- Generar casos de prueba extras.
- Correr el mismo caso dos veces seguidas para controlar inicialización.
- NO verificar a ojo la salida:
  `./a.out < input > mioutput; diff mioutput output`

Probando los programas

Tamaño de los datos

Familiarizarse con posibles tamaños de entrada o de los datos:

- 32-bit signed integers (**int**) y 64-bit signed integers (**long long**), tienen como límites sup $2^{31} - 1 \approx 2 * 10^9$ y $2^{63} - 1 \approx 9 - 10^18$ respect.
- INtegers unsigned para números no negativos 32-bit (**unsigned int**) y 64-bit unsigned integers (**unsigned long long**) con límites sup $2^{32} - 1 \approx 4 * 10^9$ y $2^{64} - 1 \approx 1{,}8 * 10^{19}$ respect.
- números $\geq 2^{64}$, usar Big Integer.

Manos a la obra!

- UVa Online Judge: http://uva.onlinejudge.org/
- Sphere Online Judge (SPOJ): http://www.spoj.com/
- COJ: http://coj.uci.cu/index.xhtml
- LiveArchive: http://livearchive.onlinejudge.org/