Last Updated: 11/10/2008 03:53:32 PM

Motivation for academia: Grouping problems into specific categories is very useful, especially for those who want to practice on a particular problem type. For example: you want to practice backtracking, you can select several problems from the backtracking list and try to solve them. You can then try to solve all variety of problems of the same type to master the technique. Those who are in process of learning a new algorithmic concept should try solving problems using this problem classification.

However, maintaining such list is very very troublesome, especially when the numbers of problems in UVa Online Judge already reach 1500+... However, there is a good news that this feature will be included as a built-in feature in UVa OJ **soon**, where once you got a problem Accepted, you got a chance to help us classify the method to solve that particular problems from a **prototype controlled list** below.

The **prototype controlled list** of solving methods, sorted based on the frequency of appearances, as well as some (max 5 per category) example problems from UVa Online Judge to help beginners in understanding this classification scheme:

1. **Ad Hoc:** Ad Hoc problems appears more often than other type of problems. In these problems, there is no textbook algorithm to solve them, but you need to think or follow problem description closely to come up with a good solution.
   Example: 460, 591, 642, 837, 10015

   Simulation: problems that can only be solved after you simulate the actual process given in the problem description, we can consider this as "Ad Hoc" too.
   Example: 100, 101, 440, 758, 10033

2. **Math:** Mathematical problems are the second highest type of problems in programming contest. Usually, specific knowledge of Number Theory or well known algorithms in Mathematics are required to solve them efficiently. Since this topic is too broad, we classify them into sub categories, sorted from the most common sub categories first.
   Example: 190, 474, 568, 725, 10071

   Math (Prime Number): Prime numbers (and their composite counterpart) occurs quite often, specific knowledge in Prime number checker, such as: advanced trial division, or Prime number generator, such as: Sieve of Eratosthenes, belong in this category.
   Example: 406, 583

   Math (Number Theory): Techniques such as Euclid, Extended_Euclid, Chinese Remainder Theorem, and other nitty gritties from

Number theories belongs to this category.
Example: 10104

Math (Computational Geometry): This category combines solution related to "pure Geometry" (such as Trigonometry solutions, Area of Triangle, etc) and "Computational Geometry" (Convex Hull, Plane Sweep, Line Intersection, etc).
Example: 10242, 10245

Math (Big Integer): Anytime a solution requires a number that exceeds long long (2^63-1), or unsigned long long if you only need positive values (2^64), you need to rely on Big Integer libraries to solve them.
Example: 10183, 10220

Math (Modular Arithmetic): Any solution that requires you to raise a number to a very high power but requires the modular fraction of it belongs to this category
Example: 374

Math (Algebra): Any solution that requires you to solve system of equations belongs to this category
Example: 10105

Math (Combinatorics): Any solution that requires you to count something belongs to this category
Example: 10007

3. **Backtracking a.k.a Complete Search a.k.a Brute force:** These three terms are actually the same. There is no efficient solution, therefore you let the computer try-them-all but prune unnecessary search space... Believe it or not, this may be one of the most popular technique to solve a problem if you can't figure out the best solution but realize that the input size is quite small or time limit is quite long...
Example: 416, 524, 10285

4. **Dynamic Programming:** Put your general DP solution to this category if it doesn't fit to any sub-categories below, otherwise, classify them as one of the following well-known DP solutions:
Example: 116, 136, 585, 640, 10003

DP (Longest Increasing Subsequence)
Example: 231, 481, 497, 10051, 10131

DP (Longest Common Subsequence)
Example: 531, 10066, 10100,10192, 10405

DP (Coin Changing)
Example: 147, 357, 674

DP (Edit Distance)

Example: 164, 526

DP (Matrix Chain Multiplication)
Example: 348

DP (Max Interval Sum)
Example: 507

DP (0-1 Knapsack)
Example: 10130

5. **Graph:** Problems that can be modeled into Graph (Trees are considered as Graph) requires either a specific well known graph algorithms or brute force solution. Put a problem in this category if there exist a well known Graph algorithms to solve them

   Graph (Traversal): standard graph problem that involves traversal (DFS/BFS)
   Example: 10009

   Graph (Single Source Shortest Path): Dijkstra, or BFS on unweighted graph
   Example: 336, 762

   Graph (All-Pairs Shortest Path): Multiple Dijkstra, Floyd Warshall or Johnson algorithm
   Example: 534, 544, 10048, 10099, 10171

   Graph (Minimum Spanning Tree): Prim or Kruskal
   Example: 10034, 10147, 10397

   Graph (Articulation Point): Finding the weakest link in the graph
   Example: 315, 796

   Graph (Flood Fill): Standard recursive flood fill on graph
   Example: 352, 572

   Graph (Network Flow): Ford Fulkerson/Edmonds Karp algorithm
   Example: 820, 10092

   Graph (Maximum Bipartite Matching): Either transform this into Network flow solution or use Hungarian matching
   Example: 670, 753

   Graph Theory: Any solutions that requires a specific properties of graph belongs to this such as Euler Cycle/Path, Euler formula to count faces of planar graph: V-E+F=2, etc...
   Example: 10596

6. **Greedy Algorithm:** Greedy, although inherently included in other algorithms such as MST, or Dijkstra... are still naturally applicable to solve many problems.
Example: 10020, 10249, 10340

7. **Divide & Conquer**: Any solutions that breaks original problem into two or more sub-problems, solve them recursively, and combines the result belongs to this category. This category usually part of other category.
Example: 374, 495

8. **Sorting:** Sorting is inherently required for solving most problems... but sometimes there exist some problems that purely test our sorting skill.... set them into this field if that is true.
Example: 299

9. **String Processing**: General string processing problem.

   Input/Output-related: Any type of solution that given input, transform it to the desired output.
   Example: 400, 403

   Recursive Descent Parser (BNF Parser): You are given a well formed grammar, and need to evaluate correctness of a language / string. Solution for this type of question usually is a RDF parser.
   Example: 464

10. **Advanced Data Structures:** There exist problems that requires advanced data structures (Fibonacci/Leftist Heaps, Sets, Dictionaries, etc)

    Set (Union-Find): There exist problems which requires testing of set
    Example: 793

    Dictionaries (Hash table, Balanced Search Trees, etc): There exist problems which are easy but requires sophisticated implementation of dictionary data structure / or you can use STL map, of advanced data structures.
    Example: 10226

11. **Binary Search:** There is a small number of problems that clearly solvable using binary search. However, if it is possible, one needs to sort the input first...
Example: 10295

12. **Not Classified Yet:** default values for all unclassified problems

On every problems that I've solved and written hints, I'll indicate their classification. The full implementation of this classification scheme will be available in UVa website soon... Stay tuned :)
