UPSILON PI EPSILON

Jeffrey L. Popyack

From Abacus to Zdravstvuyte

Zdravstvuyte!

This Russian greeting commemorates the 2013 ACM International Collegiate Programming Contest, held this summer in St. Petersburg, Russia. Upsilon Pi Epsilon was again a proud co-sponsor of the ICPC, which was also the setting for presentation of the Abacus Award to Bjarne Stroustrup.

Upsilon Pi Epsilon

Upsilon Pi Epsilon is an honor society founded at Texas A&M University in 1967. It remains the only National Honor Society for the computing and information disciplines and is recognized as such by the Association for Computing Machinery and IEEE Computer Society. Its mission is to recognize academic excellence at both the undergraduate and graduate levels in the Computing and Information Disciplines. UPE promotes the computing and information disciplines and encourages their contribution to the enhancement of knowledge. Information about its history and ideals, how to start a new chapter and much more may be found on the UPE website [6].

International Collegiate Programming Contest

This summer was witness to a spectacular display of outstanding student performance, namely the ACM-ICPC International Collegiate Programming Contest sponsored by IBM [1]. World Finals Champion St. Petersburg National Research University of Information Technologies, Mechanics and Optics (SPbNRU ITMO) took top honors for the 2nd consecutive year and 4th time in the last 6 years. With a victory in 2004 also, SPbNRU ITMO has won half the contests in the last decade. Even before the contest, their programmers had some celebrity among the other contestants, for their prowess not only in the ICPC but other coding competitions. If it had an Olympic feel, why not? As Executive Director Bill Poucher said, "Serious problems call for great minds. ... They are athletes of innovation."

Bjarne Stroustrup addresses the ACM-ICPC audience at IBM Tech Trek after accepting the UPE Abacus Award

The problem set was interesting and provocative, as usual. Dealing with such diverse topics as gambling strategies at a casino with a giveback promotion, traffic routing for drivers who are likely to ignore instructions not benefiting them directly, permutations of prime factorizations, and assembling collections of randomly ordered nesting wooden dolls (matroshkas), many skills are tapped in devising an appropriate algorithm quickly and correctly.

UPE continued its longstanding support of the contest by providing a "First Solver" award to the first team to solve each of the contest problems. This was a lively contest within itself: National Taiwan University struck first, by solving Problem F ("Low Power") in a mind-boggling 10 minutes. Yes, this turned out to be the problem solved by the most teams: 107 of the 119 teams posted solutions, including 66 in the first hour. It dealt with allocating batteries with varying power outputs among chips to optimize power allocation. Still, was this an easy problem? Take a look at the problem set [8] and let me know your strategy for picking this one out of the pile, reading it, coding and submitting a correct solution– all in 10 minutes! Problem F is shown in the appendix, along with a sample solution and commentary. If you're in for a real challenge, though, you might want to take a look at Problem G ("Map Tiles"), which was not solved by any team.

TABLE 1: FIRST SOLVERS FOR EACH PROBLEM AT THE 2013 ACM-ICPC. TIME IS IN MINUTES FROM BEGINNING OF CONTEST.

Problem	Team	Time
F - Low Power	National Taiwan University	10
D - Factors	St. Petersburg National Research University of IT, Mechanics and Optics	17
A - Self-Assembly	Ludwig-Maximilians Universität München	25
J - Pollution Solution	University of Maryland	27
C - Surely You Congest	University of Warsaw	37
H - Матрёшка	National University of Singapore	49
E - Harvard	St. Petersburg National Research University of IT, Mechanics and Optics	131
I - Pirate Chest	St. Petersburg National Research University of IT, Mechanics and Optics	157
K - Up a Tree	Shanghai Jiao Tong University	190
B - Hey, Better Bettor	The University of Tokyo	191
G - Map Tiles	*Not solved*	

That involved minimizing the number of fixed-size pages needed to publish a map of an irregularly-shaped region.

Six of the 11 contest problems saw their first solution during the first hour of the contest. The next two "first solutions" did not occur until the third hour. A look at the scoreboard [5] shows some other interesting facts, perhaps none so astounding as the fact that the third and fourth teams, the University of Tokyo and National Taiwan University, finished in a dead heat, with 8 problems solved and 1060 minutes each. "Minutes" in this case refers not only to the total time to solution of each of their 8 problems solved, but a penalty of 20 minutes for each incorrect submission for a problem they eventually solved. The statistical likelihood of this happening with a problem set including 11 problems in a 5-hour contest is, ummm… *statistically unlikely*. A studious accounting of the Top 10 finishers from the past 20 years, however showed two such ties: for 3rd place in 2007 and 2nd place in 1997. On four other occasions, two teams in the Top 10 have finished within two points of each other. In case of ties, the tiebreaker is the time of the last correct submission by each team.

Not surprisingly, SPbNRU ITMO also had the most "First Solutions," with three. They solved 10 problems, more than any other team. They also had enough time at the end to attempt Problem G 15 times – apparently it was truly inscrutable! The First Solvers are shown in Table 1.

As always, the ACM-ICPC Live Archive [2] and UVa Online Judge [7] have copies of all questions, so you can try your own hand at solving these problems. As this item goes to press in mid-August, there have not yet been any attempts of the nefarious Problem G.

Abacus Award

UPE's most prestigious award is the Abacus Award, given to "an individual who has gained international renown in the profession, and over a period of several years has provided extensive support and leadership for student-related activities in the computing and information disciplines." The 2013 Abacus Award was presented to Bjarne Stroustrup of Texas A&M University, at IBM Tech Trek, held in conjunction with the ACM-ICPC. Stroustrup is best known, of course for creating and developing the C++ programming language. He has also maintained a steady presence in the ongoing standardization efforts, of which the most recent version is C++11.

I had the pleasure of introducing Stroustrup to the audience of programmers, programming contest and IBM staff, which meant I had the daunting task of familiarizing myself with the answer to his most frequently asked question, namely "How do you pronounce 'Bjarne Stroustrup'?" Fortunately, his website [4] contains a pronunciation guide, a WAV file, and a hint about stuffing a potato down your throat. In my introduction, I asked the attendees how many of them program in C++, and immediately a wave of hands shot up, belonging to nearly everyone in the room. Of course, this was lost on the man sitting in the front row, so I asked them turn it into applause as he took the stage, where he was presented a specially-engraved abacus by UPE Executive Director Orlando Madrigal.

In his address, Stroustrup paid tribute to the many projects, people and places that inspired and shaped his career. Prominent among the people were Dennis Ritchie, Kristen Nygaard, Brian Kernighan, David Wheeler (his advisor) and Alexander Stepanov. Places included Cambridge, AT&T, Texas A&M, the SubMillimeter Observatory in Hawaii, and the Superconducting Super Collider site in Texas. After all the languages, environments, projects and products he discussed, his advice to the audience was his observation that no matter the situation, he usually spends most of his time trying to figure out what the problem is. "Once you know that, the rest is fairly obvious."

The 2013 Abacus Award was presented to Bjarne Stroustrup of Texas A&M University, at IBM Tech Trek, held in conjunction with the ACM-ICPC. Stroustrup is best known, of course for creating and developing the C++ programming language. He has also maintained a steady presence in the ongoing standardization efforts, of which the most recent version is C++11.

Accompanying Stroustrup in this supercharged environment of programming wizards and zealots showed me what it must be like to travel with a rock star. Indeed, he graciously posed for photographs and signed autographs throughout his stay as "Bjarnemania" reached its peak. Following the contest, Stroustrup visited St. Petersburg Institute for Information Technologies, Mechanics and Optics, from which he received an honorary doctorate.

2014 UPE Annual Convention

The UPE Annual Convention will be held in early 2014. UPE will again conduct this convention in cyberspace, via the Polycom Video Conferencing System, courtesy of the Graduate School of Computer and Information Science at Nova Southeastern. Please keep your eyes open for details on the agenda, registration and presenting a chapter report. I'll be looking for you!

Na zdorov 'ya!

… jp

APPENDIX

Problem F ("Low Power") was solved both very early and very often in this year's contest. Here are shown a summary of the problem statement, a sample solution and discussion.

PROBLEM STATEMENT:

Problem F ("Low Power") deals with configuring optimally a set of batteries among chips. There are n machines, each containing 2 chips. The chips each receive power from k batteries, for a total of $2nk$ batteries. The batteries' power outputs are given by the values p_i , $1 \le i \le 2nk$. A chip's power output is the minimum power output of its k batteries.

Machines perform best when both chips have the same power output. If it is not possible to allocate batteries so that all n machines have chips with the same power output, it is desirable to minimize the difference in power outputs. In particular, a meaningful specification is that all n machines have differences of d or less in the power outputs of their 2 chips. You are to determine the smallest value of d for which this is true.

The input consists of the values of n, k, and $\{p_i\}$, where $2nk \le 10^6$ and $1 \le p_i \le 10^9$. The output is simply the number d.

An example with n=3 machines, each having k=2 batteries per chip and power outputs {2, 2, 3, 6, 8, 10, 12, 13, 13, 13, 14, 14}, illustrates some of the nuances of this problem. One optimal arrangement has the batteries for the first machine with 2,3 for one chip and 2,13 for the other; the next machine with 6,13 and 8,10; and the final machine with 13,14 and 13,14. The first and third machines have chips with equal power outputs, while the second machine has a difference of 2. The output therefore is 2. There are other ways to accomplish this.

The complete problem statement and some other examples may be found at [8].

DISCUSSION:

For in-depth discussions of all the problems, I refer you to solution sketches [3] by Per Austrin of KTH Royal Institute of Technology in Sweden. Per was an ACM-ICPC contestant in the 2004 and 2005 ICPC World Finals, and has been a World Finals judge since 2008, leaving a treasury of expert commentary and insights from judges and contestants on all problems since then. Here are his remarks on Problem F:

A sample solution follows my own remarks here.

- For those scoring at home, note a slight difference in problem statements between the problem set distributed at the contest [8] and those at the online judging sites [2,7]. For the original, input consisted of a single test case, but at the judging sites, input consists of several test cases. This reflects a difference in the scoring scripts used by live judges and online judges. The time limits also differ. Otherwise, the problem is unchanged. The program shown here should work in either case.
- The difference between developing a good algorithm and writing a program sufficiently robust to perform correctly for all possible data depends in large part on reading the problem statement very carefully. Figure out what pitfalls lurk among issues the problem set hasn't pointed out, and trust the judges to assemble the most insidious set of test cases imaginable. Contest veterans know that problem limits are especially meaningful. If, as in this problem, the data values are $1 \le p_i \le 10^9$, you'd better make sure your algorithm works when you have batteries with values 1 and 10^9, and for that matter,

Problem F: Low Power

Shortest judge solution: 541 bytes. Shortest team solution (during contest): 517 bytes.

This problem is also pretty easy. We describe a way to determine if a given target difference d is possible to achieve; finding the minimum is then a simple matter of binary search.

Sort the inputs so that $p_1 \le p_2 \le \ldots p_{2nk}$. First observe that we may without loss of generality assume that in an optimal solution, the smallest outputs in each pair of chips will be of the form p_i , $p_i{+}1$ (so that the power output difference is $p_i{+}1 - p_i$).

Let us say that the first battery of a machine is the one with smallest power in the machine. Note that if the first battery of a machine is pi then by observation above we can assume that the power difference of that machine is $p_i{+}1 - p_i$.

Now consider the machines sorted in increasing order of power of their first battery. Clearly, the first machine has power difference $p_2 - p_1$. For the second machine, the first battery can be any one of $p_3, \ldots, p_2k{+}1$. We then greedily choose the first $i^* \ge 3$ such that $p_{i^*}{+}1 - p_{i^*} \le d$, and use this for the second battery (if no such $i^* \le 2k + 1$ exists, there is no solution). Then we look at the third machine. The first battery of this can be either of $p_i{*}{+}2, \ldots, p_4k{+}1$, and we again greedily choose the first one resulting in a power output smaller than d. We continue this process until all batteries have been assigned.

when n=1 and k=1. If your program fails on one test case, it might as well be riddled with syntax errors.

- The use of binary search might not have come to mind immediately, but recognize that in the example, the data values could very well be on the scale of 20000000, 30000000, etc. instead of integers less than 15. In that case, a linear search through the solution space will take far too long. If you exceed the time limit, you won't know whether you just need a little more time, or if you're in an infinite loop. Per's notes about a puzzling timeout in a submission from the University of Tokyo for Problem B are well worth reading to gain further insights into the types of issues that can arise at the boundaries of a problem.

Remember, this was the easy one! The code that follows is my attempt to produce something that will satisfy three separate audiences: the online judges, CS professors who expect good use of indentation and comments, and seasoned contestants who hate wasting space on such things on good mnemonics and, well, indentation and comments. I tried to leave all three equally dissatisfied, but eventually capitulated to the online judge. **lr**

```
//-------------------------------------------------------------
// Solution for Problem F (Low Power), ICPC 2013.
// JL Popyack
//-------------------------------------------------------------

#include <iostream>
#include <vector>
#include <algorithm>
using namespace std ;

bool solvable(int d, int n, int k, vector<int>& p)
{
        //---------------------------------------------------------
        // Determine if the problem can be solved with a difference
        // of power outputs of size d or less.
        //---------------------------------------------------------

        int i=0 ;

        // p[1]-p[0] is the power difference in first machine
        bool found= ( p[1]-p[0] <= d ) ;

        //---------------------------------------------------------
        // Search the batteries that can be the first battery
        // for machine m, and find the first one that can offer
        // a gap of the desired size.
        //---------------------------------------------------------
        for(int m=2; found && (m<=n) ; m++)
        {
                int upper = (m-1)*2*k ;
                i += 2 ;
                found = false ;
                while( !found && (i<=upper) )
                {
                        found = ( p[i+1]-p[i] <= d ) ;
                        if(!found)
                                i++ ;
                }
        }
        return found ;
}

int main(void)
{
        int n ;   // number of machines
        int k ;   // number of batteries per chip

        while(cin >> n >> k)
        {
            vector<int> p(2*n*k) ;

            for(int i=0; i<2*n*k; i++)
                    cin >> p[i] ;

            sort(p.begin(),p.end()) ;

            int lo = 0 ;
            int hi = 1E9 ;  // largest possible battery size

            //---------------------------------------------------------
            // Find the smallest value for which the problem is solvable,
            // using binary search.
            //---------------------------------------------------------
            while( lo < hi )
            {
                    int mid = (lo + hi)/2 ;
                    if( solvable(mid,n,k,p) )
                            hi = mid ;
                    else
                            lo = mid + 1 ;
            }

            cout << lo << endl ;
        }

        return 0 ;
}
```


References

[1] ACM-ICPC International Collegiate Programming Contest Web Site sponsored by IBM, http://icpc.baylor.edu/
[2] ACM-ICPC Live Archive, http://livearchive.onlinejudge.org/
[3] Austrin, P. ACM ICPC World Finals 2013 Solution Sketches, http://www.csc.kth.se/~austrin/icpc/finals2013solutions.pdf
[4] Bjarne Stroustrup's homepage, http://www.stroustrup.com/
[5] Final standings in 37th Annual World Finals of the ACM International Collegiate Programming Contest, http://icpc.baylor.edu/public/worldMap/World-Finals-2013
[6] Upsilon Pi Epsilon website, http://upe.acm.org/
[7] UVa Online Judge, Universidad de Valladolid, http://uva.onlinejudge.org/
[8] 2013 ACM-ICPC World Finals Problems, http://icpc.baylor.edu/download/worldfinals/problems/icpc2013.pdf

Jeffrey L. Popyack
Department of Computer Science
Drexel University
Philadelphia, Pennsylvania 19104-2875
USA
http://www.cs.drexel.edu/~jpopyack
JPopyack@CS.Drexel.edu

DOI: 10.1145/2537753.2537765