P versus NP for Algorithm Design and Evaluation

Mirzakhmet Syzdykov
mspmail598@gmail.com
Satbayev University, Almaty, Kazakhstan

ABSTRACT

We present the difference of the “P versus NP” problem for polynomial and non-polynomial classes on the example of two contest problems held by ACM ICPC NEERC in 2004 and 2005.

INTRODUCTION

We give concise and clear comparison of P (polynomial) and NP (non-polynomial) complexity classes on the example of two problems [1, 2, 3].
We also show that solution is feasible within NP-complete problem if the number of elements in input data is very low (< 20) as per bit set definition so that $2^N < 10^9$, which is measured in performed as one second of processor operation cycle.
We will consider following problems to compare P and NP complexities:

- NP-complete: Problem "Box" [4];
- P-complete: Problem "Exploring Pyramids" [5].

The "P versus NP" theorem, which is still not proved, was formulated by Stephen Cook [2] and implies the relation between these two classes of complexity.

P AND NP CLASSES OF COMPUTATIONAL COMPLEXITY

The problem "Box" is NP-complete, however, as the number of elements in input stream is less than 6, it can be solved in almost polynomial time:

```
const int N = 6;

...

bool check (int *P, int bitmask) {
    for (int i = 0; i < N; i++)
        for (int j = 0; j < N; j++)
            if (F[i][j][0] != -1)
                if (A[P[i]][F[i][j][0] ^ ((bitmask >> i) & 1)]
                  != A[P[j]][F[i][j][1] ^ ((bitmask >> j) & 1)])
                    return false;
    return true;
}

void solve() {
        int P[N];

        for (int i = 0; i < N; ++i) {
            P[i] = i;
        }
```

```
        do {
            for (int i = 0; i < (1L << N); ++i) {
                if (check(P, i)) {
                    puts ("POSSIBLE");

                    return;
                }
            }
        } while (next_permutation(P, P + N));

        puts ("IMPOSSIBLE");
}
```

Solution for this problem is P-complete, where N = 6 and N! << 10^8, when the complexity, however is O (2^N * N! * N^3), which is also much lower than average time of running the program in few seconds on modern hardware.
And for the problem "Exploring Pyramids" [5], we use case marks in the global array in order to save the time as per each case, when input is given in single file. We use the dynamic programming approach with memorization.
The complexity of this problem, thus, is also polynomial and is defined in big-O notation as O (N^3).
The code for this problem is as follows:

```
const int N_MAX = 400;

typedef long long ll_t;

int S[N_MAX];
int P[N_MAX][N_MAX];
ll_t R[N_MAX][N_MAX];

...

ll_t F(int l, int r) {
  if (l == r) return 1;

  if (P[l][r] == CaseNumber) return R[l][r];

  P[l][r] = CaseNumber, R[l][r] = 0;

  for (int i = l + 1; i <= r; i++)
    if (S[i] == S[l])
      R[l][r] = (R[l][r] + F(l + 1, i - 1) * F(i, r)) % 1000000000;

  return R[l][r];
}
```

The code for both problems can be obtained from repository [6].

CONCLUSION

Thus, we have devised that even NP-complete problems can be solved exactly with respect to the computational volume of the state space without using dynamic programming, as when the complexity fits into this volume.

REFERENCES

1. Fortnow L. The status of the P versus NP problem //Communications of the ACM. – 2009. – Т. 52. – №. 9. – С. 78-86.
2. Cook S. The importance of the P versus NP question //Journal of the ACM (JACM). – 2003. – Т. 50. – №. 1. – С. 27-29.
3. Aaronson S., Is P. Is P versus NP formally independent? //Bulletin of the EATCS. – 2003. – Т. 81. – №. 109-136. – С. 70.
4. Problem “Box” // UVA – 1587 // https://vjudge.net/problem/UVA-1587 (accessed 11/11/2022).
5. Problem “Exploring Pyramids” // UVA – 1362 // https://vjudge.net/problem/UVA-1362 (accessed 11/11/2022).
6. Syzdykov M. ACM Problem Solutions // https://github.com/mirzakhmets/ACM (accessed 11/11/2022).