

Heap Thrust Reference 2026

ICPC Reference (Date 1 version)

Team Members:

Enrique Job Calderón (ksobrenat32)
Luis Daniel Salazar (luisdakan)
Gustavo Valenzuela (GusTimeTraveler)

Institution:

Facultad de Ingeniería, UNAM - CPCFI

December 1, 2025

Contents

1 Python stuff	2		
1.1 Regular input	2		
1.2 Fast I/O (If we use this, we're down bad)	2		
2 Graph Theory	2		
2.1 Depth First Search (DFS)	2		
2.2 Breadth First Search (BFS)	2		
3 Data Structures	2		
3.1 Segment Tree	2		
3.2 Fenwick Tree (BIT)	3		
4 Range Queries	3		
4.1 Sparse Table	3		
5 Mathematics	4		
5.1 Number Theory	4		
5.1.1 GCD and LCM	4		
5.1.2 Extended Euclidean Algorithm	4		
5.1.3 Modular Arithmetic	4		
5.2 Combinatorics	5		
5.2.1 Binomial Coefficients	5		
5.2.2 Catalan Numbers	5		
5.3 Probability	5		
5.3.1 Basic Formulas	5		
5.3.2 Expected Value	5		
5.4 Algebra	5		
5.4.1 Matrix Operations	5		
5.4.2 System of Linear Equations	6		
5.5 Calculus	6		
5.5.1 Derivatives	6		
5.5.2 Integration	6		
5.5.3 Series	6		
6 String Algorithms	6		
6.1 KMP (Knuth-Morris-Pratt)	6		
6.2 Z-Algorithm	7		
7 Computational Geometry	7		
7.1 Basic Formulas	7		
7.1.1 Distance and Dot Product	7		
7.1.2 Cross Product	7		
7.2 Convex Hull	7		
8 Dynamic Programming	8		
8.1 Longest Increasing Subsequence	8		
8.2 Knapsack	8		

1 Python stuff

1.1 Regular input

```

1 s = input() # This is a string
2 n = int(input()) # This is an integer
3
4 arr = input().split() # This is an array of strings
5
6 # Array of integers (list comprehension)
7 a = [int(x) for x in input().split()]
8 x,y,z = [int(i) for i in input().split()] # Multiple integers

```

1.2 Fast I/O (If we use this, we're down bad)

```

1 import sys
2 input = sys.stdin.readline
3 n = 220
4 sys.stdout.write(str(n) + '\n')

```

2 Graph Theory

2.1 Depth First Search (DFS)

Example implementation:

```

1 // Depth First Search
2 vector<int> adj[MAXN];
3 bool visited[MAXN];
4
5 void dfs(int u) {
6     visited[u] = true;
7     for (int v : adj[u]) {
8         if (!visited[v]) {
9             dfs(v);
10        }
11    }
12 }

```

Listing 1: DFS Implementation

2.2 Breadth First Search (BFS)

Example implementation:

```

1 // Breadth First Search
2 vector<int> adj[MAXN];
3 int dist[MAXN];
4
5 void bfs(int start) {
6     queue<int> q;
7     memset(dist, -1, sizeof(dist));
8     dist[start] = 0;
9     q.push(start);
10
11     while (!q.empty()) {
12         int u = q.front();
13         q.pop();
14
15         for (int v : adj[u]) {
16             if (dist[v] == -1) {
17                 dist[v] = dist[u] + 1;
18                 q.push(v);
19             }
20         }
21     }
22 }

```

Listing 2: BFS Implementation

3 Data Structures

3.1 Segment Tree

Example implementation:

```

1 // Segment Tree for Range Sum Query
2 class SegmentTree {
3     vector<long long> tree;
4     int n;
5
6 public:
7     SegmentTree(vector<int>& arr) {
8         n = arr.size();
9         tree.resize(4 * n);
10        build(arr, 0, 0, n - 1);
11    }
12
13    void build(vector<int>& arr, int node, int start, int end) {
14        if (start == end) {

```

```

15     tree[node] = arr[start];
16 } else {
17     int mid = (start + end) / 2;
18     build(arr, 2*node+1, start, mid);
19     build(arr, 2*node+2, mid+1, end);
20     tree[node] = tree[2*node+1] + tree[2*node+2];
21 }
22 }
23
24 void update(int node, int start, int end, int idx, int val) {
25     if (start == end) {
26         tree[node] = val;
27     } else {
28         int mid = (start + end) / 2;
29         if (idx <= mid) {
30             update(2*node+1, start, mid, idx, val);
31         } else {
32             update(2*node+2, mid+1, end, idx, val);
33         }
34         tree[node] = tree[2*node+1] + tree[2*node+2];
35     }
36 }
37
38 long long query(int node, int start, int end, int l, int r) {
39     if (r < start || end < l) return 0;
40     if (l <= start && end <= r) return tree[node];
41     int mid = (start + end) / 2;
42     return query(2*node+1, start, mid, l, r) +
43            query(2*node+2, mid+1, end, l, r);
44 }
45
46 void update(int idx, int val) { update(0, 0, n-1, idx, val); }
47 long long query(int l, int r) { return query(0, 0, n-1, l, r); }
48 };

```

Listing 3: Segment Tree

3.2 Fenwick Tree (BIT)

Example implementation:

```

1 // Fenwick Tree (Binary Indexed Tree)
2 class FenwickTree {
3     vector<long long> tree;
4     int n;
5
6 public:
7     FenwickTree(int n) : n(n), tree(n + 1, 0) {}
8
9     void update(int idx, int delta) {
10         for (++idx; idx <= n; idx += idx & -idx) {

```

```

11             tree[idx] += delta;
12         }
13     }
14
15     long long query(int idx) {
16         long long sum = 0;
17         for (++idx; idx > 0; idx -= idx & -idx) {
18             sum += tree[idx];
19         }
20         return sum;
21     }
22
23     long long query(int l, int r) {
24         return query(r) - (l > 0 ? query(l - 1) : 0);
25     }
26 };

```

Listing 4: Fenwick Tree

4 Range Queries

4.1 Sparse Table

Example implementation:

```

1 // Sparse Table for Range Minimum Query
2 class SparseTable {
3     vector<vector<int>> table;
4     vector<int> log;
5     int n;
6
7 public:
8     SparseTable(vector<int>& arr) {
9         n = arr.size();
10        int maxLog = log2(n) + 1;
11        table.assign(n, vector<int>(maxLog));
12        log.assign(n + 1, 0);
13
14        // Precompute logarithms
15        for (int i = 2; i <= n; i++) {
16            log[i] = log[i/2] + 1;
17        }
18
19        // Build sparse table
20        for (int i = 0; i < n; i++) {
21            table[i][0] = arr[i];
22        }
23
24        for (int j = 1; j < maxLog; j++) {

```

```

25         for (int i = 0; i + (1 << j) <= n; i++) {
26             table[i][j] = min(table[i][j-1],
27                               table[i + (1 << (j-1))][j-1]);
28         }
29     }
30 }
31
32 int query(int l, int r) {
33     int j = log[r - l + 1];
34     return min(table[l][j], table[r - (1 << j) + 1][j]);
35 }
36 };

```

Listing 5: Sparse Table

```

4     if (b == 0) {
5         x = 1;
6         y = 0;
7         return a;
8     }
9     int x1, y1;
10    int g = extgcd(b, a % b, x1, y1);
11    x = y1;
12    y = x1 - (a / b) * y1;
13    return g;
14 }

```

Listing 7: Extended GCD

5 Mathematics

5.1 Number Theory

5.1.1 GCD and LCM

$$\gcd(a, b) = \gcd(b, a \bmod b)$$

$$\text{lcm}(a, b) = \frac{ab}{\gcd(a, b)}$$

```

1 // Greatest Common Divisor
2 int gcd(int a, int b) {
3     return b == 0 ? a : gcd(b, a % b);
4 }
5
6 int lcm(int a, int b) {
7     return a / gcd(a, b) * b;
8 }

```

Listing 6: GCD Implementation

5.1.2 Extended Euclidean Algorithm

For $ax + by = \gcd(a, b)$:

```

1 // Extended Euclidean Algorithm
2 // Returns gcd(a, b) and finds x, y such that ax + by = gcd(a, b)
3 int extgcd(int a, int b, int &x, int &y) {

```

5.1.3 Modular Arithmetic

$a^{-1} \bmod m$ exists iff $\gcd(a, m) = 1$

$a^{\phi(m)} \equiv 1 \pmod{m}$ (Euler's theorem)

$a^{p-1} \equiv 1 \pmod{p}$ (Fermat's little theorem)

```

1 // Modular Exponentiation
2 long long mod_pow(long long base, long long exp, long long mod) {
3     long long result = 1;
4     base %= mod;
5     while (exp > 0) {
6         if (exp & 1) result = (result * base) % mod;
7         base = (base * base) % mod;
8         exp >>= 1;
9     }
10    return result;
11 }
12
13 // Modular Inverse using Fermat's Little Theorem
14 // Works when mod is prime
15 long long mod_inv(long long a, long long mod) {
16     return mod_pow(a, mod - 2, mod);
17 }

```

Listing 8: Modular Exponentiation

5.2 Combinatorics

5.2.1 Binomial Coefficients

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

$$\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$$

$$\sum_{k=0}^n \binom{n}{k} = 2^n$$

```

1 // Binomial Coefficients with Modular Arithmetic
2 const int MOD = 1e9 + 7;
3 const int MAXN = 1e6 + 5;
4 long long fact[MAXN], inv_fact[MAXN];
5
6 long long mod_pow(long long base, long long exp, long long mod) {
7     long long result = 1;
8     while (exp > 0) {
9         if (exp & 1) result = (result * base) % mod;
10        base = (base * base) % mod;
11        exp >>= 1;
12    }
13    return result;
14 }
15
16 void precompute() {
17     fact[0] = 1;
18     for (int i = 1; i < MAXN; i++) {
19         fact[i] = (fact[i-1] * i) % MOD;
20     }
21     inv_fact[MAXN-1] = mod_pow(fact[MAXN-1], MOD - 2, MOD);
22     for (int i = MAXN-2; i >= 0; i--) {
23         inv_fact[i] = (inv_fact[i+1] * (i+1)) % MOD;
24     }
25 }
26
27 long long nCr(int n, int r) {
28     if (r < 0 || r > n) return 0;
29     return (fact[n] * inv_fact[r] % MOD) * inv_fact[n-r] % MOD;
30 }

```

Listing 9: Binomial Coefficients

5.2.2 Catalan Numbers

$$C_n = \frac{1}{n+1} \binom{2n}{n} = \frac{(2n)!}{(n+1)!n!}$$

$$C_n = \sum_{i=0}^{n-1} C_i C_{n-1-i}$$

$$C_0 = 1, \quad C_{n+1} = \frac{2(2n+1)}{n+2} C_n$$

5.3 Probability

5.3.1 Basic Formulas

$$P(A \cup B) = P(A) + P(B) - P(A \cap B)$$

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

$$P(A \cap B) = P(A|B) \cdot P(B) = P(B|A) \cdot P(A)$$

5.3.2 Expected Value

$$E[X] = \sum_i x_i \cdot P(X = x_i)$$

$$E[aX + b] = aE[X] + b$$

$$E[X + Y] = E[X] + E[Y]$$

5.4 Algebra

5.4.1 Matrix Operations

Matrix multiplication complexity: $O(n^3)$

$$(AB)_{ij} = \sum_{k=1}^n A_{ik} B_{kj}$$

$$(AB)^T = B^T A^T$$

$$\det(AB) = \det(A) \det(B)$$

5.4.2 System of Linear Equations

Gaussian elimination: $O(n^3)$

$$Ax = b$$

If $\det(A) \neq 0$, unique solution exists

5.5 Calculus

5.5.1 Derivatives

$$(x^n)' = nx^{n-1}$$

$$(e^x)' = e^x$$

$$(\ln x)' = \frac{1}{x}$$

$$(\sin x)' = \cos x$$

$$(\cos x)' = -\sin x$$

5.5.2 Integration

$$\int x^n dx = \frac{x^{n+1}}{n+1} + C \quad (n \neq -1)$$

$$\int e^x dx = e^x + C$$

$$\int \frac{1}{x} dx = \ln |x| + C$$

$$\int \sin x dx = -\cos x + C$$

$$\int \cos x dx = \sin x + C$$

5.5.3 Series

$$\sum_{i=1}^n i = \frac{n(n+1)}{2}$$

$$\sum_{i=1}^n i^2 = \frac{n(n+1)(2n+1)}{6}$$

$$\sum_{i=0}^n x^i = \frac{1-x^{n+1}}{1-x} \quad (x \neq 1)$$

6 String Algorithms

6.1 KMP (Knuth-Morris-Pratt)

Example implementation:

```

1 // Knuth-Morris-Pratt Algorithm
2 vector<int> computeLPS(string pattern) {
3     int m = pattern.length();
4     vector<int> lps(m);
5     int len = 0;
6     lps[0] = 0;
7
8     for (int i = 1; i < m; ) {
9         if (pattern[i] == pattern[len]) {
10             len++;
11             lps[i] = len;
12             i++;
13         } else {
14             if (len != 0) {
15                 len = lps[len - 1];
16             } else {
17                 lps[i] = 0;
18                 i++;
19             }
20         }
21     }
22     return lps;
23 }
24
25 vector<int> KMP(string text, string pattern) {
26     vector<int> lps = computeLPS(pattern);
27     vector<int> matches;
28     int n = text.length();
29     int m = pattern.length();
30     int i = 0, j = 0;

```

```

31
32 while (i < n) {
33     if (text[i] == pattern[j]) {
34         i++;
35         j++;
36     }
37
38     if (j == m) {
39         matches.push_back(i - j);
40         j = lps[j - 1];
41     } else if (i < n && text[i] != pattern[j]) {
42         if (j != 0) {
43             j = lps[j - 1];
44         } else {
45             i++;
46         }
47     }
48 }
49 return matches;
50 }

```

Listing 10: KMP Algorithm

6.2 Z-Algorithm

Example implementation:

```

1 // Z-Algorithm
2 vector<int> z_function(string s) {
3     int n = s.length();
4     vector<int> z(n);
5     int l = 0, r = 0;
6
7     for (int i = 1; i < n; i++) {
8         if (i <= r) {
9             z[i] = min(r - i + 1, z[i - l]);
10        }
11        while (i + z[i] < n && s[z[i]] == s[i + z[i]]) {
12            z[i]++;
13        }
14        if (i + z[i] - 1 > r) {
15            l = i;
16            r = i + z[i] - 1;
17        }
18    }
19    return z;
20 }
21
22 // Pattern matching using Z-algorithm
23 vector<int> patternMatch(string text, string pattern) {
24     string combined = pattern + "$" + text;

```

```

25     vector<int> z = z_function(combined);
26     vector<int> matches;
27     int m = pattern.length();
28
29     for (int i = m + 1; i < combined.length(); i++) {
30         if (z[i] == m) {
31             matches.push_back(i - m - 1);
32         }
33     }
34     return matches;
35 }

```

Listing 11: Z-Algorithm

7 Computational Geometry

7.1 Basic Formulas

7.1.1 Distance and Dot Product

$$d(P_1, P_2) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

$$\vec{a} \cdot \vec{b} = a_x b_x + a_y b_y = |\vec{a}| |\vec{b}| \cos \theta$$

7.1.2 Cross Product

$$\vec{a} \times \vec{b} = a_x b_y - a_y b_x$$

$$|\vec{a} \times \vec{b}| = |\vec{a}| |\vec{b}| \sin \theta$$

7.2 Convex Hull

Example implementation:

```

1 // Convex Hull using Graham Scan
2 struct Point {
3     long long x, y;
4     bool operator<(const Point& p) const {
5         return x < p.x || (x == p.x && y < p.y);
6     }
7 };
8
9 long long cross(Point O, Point A, Point B) {

```



```

10     return (A.x - O.x) * (B.y - O.y) - (A.y - O.y) * (B.x - O.x);
11 }
12
13 vector<Point> convexHull(vector<Point> points) {
14     int n = points.size(), k = 0;
15     if (n <= 3) return points;
16
17     sort(points.begin(), points.end());
18     vector<Point> hull(2 * n);
19
20     // Build lower hull
21     for (int i = 0; i < n; i++) {
22         while (k >= 2 && cross(hull[k-2], hull[k-1], points[i]) <=
23             0) k--;
24         hull[k++] = points[i];
25     }
26
27     // Build upper hull
28     for (int i = n - 2; i >= 0; i--) {
29         while (k >= 2 && cross(hull[k-2], hull[k-1], points[i]) <=
30             0) k--;
31         hull[k++] = points[i];
32     }
33
34     hull.resize(k - 1);
35     return hull;
36 }

```

Listing 12: Convex Hull (Graham Scan)

```

14     return dp.size();
15 }
16
17 // If you need to reconstruct the sequence
18 vector<int> lisSequence(vector<int>& arr) {
19     int n = arr.size();
20     vector<int> dp, parent(n, -1), indices;
21
22     for (int i = 0; i < n; i++) {
23         auto it = lower_bound(dp.begin(), dp.end(), arr[i]);
24         int pos = it - dp.begin();
25
26         if (it == dp.end()) {
27             dp.push_back(arr[i]);
28             indices.push_back(i);
29         } else {
30             *it = arr[i];
31             indices[pos] = i;
32         }
33
34         if (pos > 0) {
35             parent[i] = indices[pos - 1];
36         }
37     }
38
39     vector<int> result;
40     int curr = indices.back();
41     while (curr != -1) {
42         result.push_back(arr[curr]);
43         curr = parent[curr];
44     }
45     reverse(result.begin(), result.end());
46     return result;
47 }

```

Listing 13: LIS in $O(n \log n)$

8 Dynamic Programming

8.1 Longest Increasing Subsequence

Example implementation:

```

1 // Longest Increasing Subsequence in  $O(n \log n)$ 
2 int lis(vector<int>& arr) {
3     vector<int> dp;
4
5     for (int x : arr) {
6         auto it = lower_bound(dp.begin(), dp.end(), x);
7         if (it == dp.end()) {
8             dp.push_back(x);
9         } else {
10             *it = x;
11         }
12     }
13 }

```

8.2 Knapsack

Example implementation:

```

1 // 0-1 Knapsack Problem
2 int knapsack(int W, vector<int>& weights, vector<int>& values) {
3     int n = weights.size();
4     vector<vector<int>> dp(n + 1, vector<int>(W + 1, 0));
5
6     for (int i = 1; i <= n; i++) {
7         for (int w = 0; w <= W; w++) {
8             dp[i][w] = dp[i-1][w];
9             if (weights[i-1] <= w) {
10                 dp[i][w] = max(dp[i][w],

```

```
11         dp[i-1][w - weights[i-1]] + values[i-1]);
12     }
13 }
14 }
15
16     return dp[n][W];
17 }
18
19 // Space-optimized version
20 int knapsackOptimized(int W, vector<int>& weights, vector<int>& values) {
21     int n = weights.size();
22     vector<int> dp(W + 1, 0);
23
24     for (int i = 0; i < n; i++) {
25         for (int w = W; w >= weights[i]; w--) {
26             dp[w] = max(dp[w], dp[w - weights[i]] + values[i]);
27         }
28     }
29
30     return dp[W];
31 }
```

Listing 14: 0-1 Knapsack