

Contents

1 Basic	1	7 Data Structure	23
1.1 default code	1	7.1 Link-Cut Tree	23
1.2 .vimrc	1	7.2 Black Magic	23
1.3 Increase Stack Size (linux)	1	8 Others	24
1.4 Misc	1	8.1 SOS dp	24
1.5 check	1	8.2 Number of Occurrences of Digit	24
2 flow	1	8.3 Find max tangent(x,y is increasing)	24
2.1 ISAP	1	8.4 Exact Cover Set	24
2.2 MinCostFlow	1	8.5 Hilbert Curve	24
2.3 Dinic	2	8.6 De Bruijn sequence	24
2.4 Kuhn Munkres 最大完美二分匹配	2		
2.5 Directed MST	2		
2.6 SW min-cut (不限 S-T 的 min-cut)	3		
2.7 Max flow with lower/upper bound	3		
2.8 Flow Method	3		
3 Math	3		
3.1 FFT	3		
3.2 NTT	3		
3.3 Fast Walsh Transform	4		
3.4 Poly operator	4		
3.5 O(1)mul	5		
3.6 Linear Recurrence	5		
3.7 Miller Rabin	5		
3.8 Faulhaber ($\sum_{i=1}^n i^p$)	5		
3.9 Chinese Remainder	6		
3.10 Pollard Rho	6		
3.11 Josephus Problem	6		
3.12 ax+by=gcd	6		
3.13 Discrete sqrt	6		
3.14 Romberg 定積分	6		
3.15 Prefix Inverse	6		
3.16 Roots of Polynomial 找多項式的根	6		
3.17 Primes	7		
3.18 Phi	7		
3.19 Result	7		
4 Geometry	7		
4.1 definition	7		
4.2 Intersection of 2 lines	7		
4.3 halfPlaneIntersection	8		
4.4 Convex Hull	8		
4.5 Convex Hull 3D	8		
4.6 Intersection of 2 segments	9		
4.7 Intersection of circle and segment	9		
4.8 Intersection of polygon and circle	9		
4.9 Point In Polygon	9		
4.10 Intersection of 2 circles	9		
4.11 Circle cover	9		
4.12 Convex Hull trick	10		
4.13 Tangent line of two circles	10		
4.14 Minimum distance of two convex	11		
4.15 KD Tree	11		
4.16 Poly Union	11		
4.17 Lower Concave Hull	12		
4.18 Min Enclosing Circle	12		
4.19 Min Enclosing Ball	12		
4.20 Minkowski sum	12		
4.21 Li Chao Segment Tree	13		
4.22 Area of Rectangles	13		
4.23 Min dist on Cuboid	13		
4.24 Heart of Triangle	13		
5 Graph	14		
5.1 DominatorTree	14		
5.2 Maximum Clique 最大團	14		
5.3 Maximal Clique 極大團	14		
5.4 Centroid Decomposition	15		
5.5 Strongly Connected Component	15		
5.6 Dynamic MST	15		
5.7 Maximum General graph Matching	16		
5.8 Minimum General Weighted Matching	16		
5.9 Maximum General Weighted Matching	16		
5.10 Minimum Steiner Tree	18		
5.11 BCC based on vertex	18		
5.12 Min Mean Cycle	18		
5.13 Directed Graph Min Cost Cycle	19		
5.14 K-th Shortest Path	19		
5.15 SPFA	20		
5.16 差分約束	20		
6 String	20		
6.1 PalTree	20		
6.2 KMP	21		
6.3 SAIS	21		
6.4 SuffixAutomata	21		
6.5 Z Value	22		
6.6 BWT	22		
6.7 ZValue Palindrome	22		
6.8 Smallest Rotation	22		
6.9 Cyclic LCS	22		
		1.1 default code	
		<code>#ifdef LOCAL // ===== Local =====</code>	
		<code>void dbg() { cerr << '\n'; }</code>	
		<code>template<class T, class ...U> void dbg(T a, U ...b) {</code>	
		<code>cerr << a << ' ', dbg(b...); }</code>	
		<code>template<class T> void org(T l, T r) { while (l != r)</code>	
		<code>cerr << *l++ << ' '; cerr << '\n'; }</code>	
		<code>#define debug(args...) (dbg("#> (" + string(#args) + ")</code>	
		<code>= ("", args, ""))</code>	
		<code>#define orange(args...) (cerr << "#> [" + string(#args)</code>	
		<code>+ ") = ", org(args))</code>	
		<code>#else // ===== OnlineJudge =====</code>	
		<code>#pragma GCC optimize("O3,unroll-loops")</code>	
		<code>#pragma GCC target("avx2,bmi,bmi2,lzcnt,popcnt")</code>	
		<code>#define debug(...) ((void)0)</code>	
		<code>#define orange(...) ((void)0)</code>	
		<code>#endif</code>	
		<code>template<class T> bool chmin(T &a, T b) { return b < a</code>	
		<code>and (a = b, true); }</code>	
		<code>template<class T> bool chmax(T &a, T b) { return b > a</code>	
		<code>and (a = b, true); }</code>	
		1.2 .vimrc	
		<code>set nu rnu ts=4 sw=4 bs=2 ai hls cin mouse=a</code>	
		<code>color default</code>	
		<code>sy on</code>	
		<code>inoremap {<CR> {<CR>}<C-o>0</code>	
		<code>inoremap jk <Esc></code>	
		<code>nnoremap J 5j</code>	
		<code>nnoremap K 5k</code>	
		<code>nnoremap run :w<bar>!g++ -std=c++14 -DLOCAL -Wfatal-</code>	
		<code>errors -o test "%> " && echo "done." && time ./test<</code>	
		<code>CR></code>	
		1.3 Increase Stack Size (linux)	
		<code>#include <sys/resource.h></code>	
		<code>void increase_stack_size() {</code>	
		<code>const rlim_t ks = 64*1024*1024;</code>	
		<code>struct rlimit rl;</code>	
		<code>int res=getrlimit(RLIMIT_STACK, &rl);</code>	
		<code>if(res==0){</code>	
		<code>if(rl.rlim_cur<ks){</code>	
		<code>rl.rlim_cur=ks;</code>	
		<code>res=setrlimit(RLIMIT_STACK, &rl);</code>	
		<code>} }</code>	
		1.4 Misc	
		編譯參數: -std=c++14 -Wall -Wshadow (-fsanitize=	
		undefined)	
		mt19937 gen(chrono::steady_clock::now().	
		time_since_epoch().count());	
		int randint(int lb, int ub)	
		{ return uniform_int_distribution<int>(lb, ub)(gen); }	
		#define SECS ((double)clock() / CLOCKS_PER_SEC)	
		struct KeyHasher {	
		size_t operator()(const Key& k) const {	
		return k.first + k.second * 100000;	
		} };	
		typedef unordered_map<Key,int,KeyHasher> map_t;	
		__builtin_popcountll // 二進位有幾個1	
		__builtin_clzll // 左起第一個1之前0的個數	
		__builtin_parityll // 1的個數的奇偶性	
		__builtin_mul_overflow(a,b,&h) // a*b是否溢位	