Competitive Programming

Bjarki Ágúst Guðmundsson Trausti Sæmundsson Ingólfur Eðvarðsson

October 28, 2012

Contents

1 Data Structures

1.1 Union-Find

```
void uf_init(int* arr, int n) {
	for (int i = 0; i < n; i++) {
		arr[i] = i;
	}
}

int uf_find(int* arr, int i) {
	return arr[i] == i ? i : (arr[i] = uf_find(arr, arr[i]));
}

void uf_union(int* arr, int i, int j) {
	arr[uf_find(arr, i)] = uf_find(uf_find(arr, j));
}
```

1.2 Segment Tree

1.3 Fenwick Tree

1.4 Interval Tree

2 Graphs

2.1 Breadth-First Search

An example of a breadth-first search that counts the number of edges on the shortest path from the starting vertex to the ending vertex. Note that it assumes that the two vertices are connected.

```
int bfs(int start, int end, vector<vi> adj_list) {
	queue<ii> Q;
	Q.push(ii(start, 0));

	while (true) {
		ii cur = Q.top(); Q.pop();

		if (cur.first == end) {
			return cur.second;
		}

		vi& adj = adj_list[cur.first];
		for (vi::iterator it = adj.begin(); it != adj.end(); it++) {
			Q.push(ii(*it, cur.second + 1));
		}
	}
}
```

2.2 Depth-First Search

2.3 Single Source Shortest Path

2.3.1 Dijkstra's algorithm

```
#define MAXEDGES 20000
bool done[MAXEDGES];

int dijkstra(int start, int end, vvii& adj_list) {
        memset(done, 0, MAXEDGES);
        priority_queue<ii, vii, greater<ii> > pq;
        pq.push(ii(0, start));

        while (!pq.empty()) {
                ii current = pq.top(); pq.pop();
                done[current.second] = true;

                if (current.second == end)
                        return current.first;

                vii &vtmp = adj_list[current.second];
                for (vii::iterator it=vtmp.begin(); it != vtmp.end(); it++)
                        if (!done[it->second])
                                pq.push(ii(current.first + it->first,
                                   it->second));
        }
        return -1;
}
```

2.3.2 Bellman-Ford algorithm

2.4 All Pairs Shortest Path

2.4.1 Floyd-Warshall algorithm

2.5 Connected Components

2.5.1 Modified Breadth-First Search

2.6 Strongly Connected Components

2.6.1 Kosaraju's algorithm

2.6.2 Tarjan's algorithm

2.7 Topological Sort

2.7.1 Modified Breadth-First Search

2.8 Articulation Points/Bridges

2.8.1 Modified Depth-First Search

3 Number Theory

3.1 Binomial Coefficients

The binomial coefficient $\binom{n}{k} = \frac{n!}{k!(n-k)!}$ is the number of ways to choose k items out of a total of n items.

```
int factorial(int n) {
        int res = 1;
        while (n) {
                res *= n--;
        }

        return res;
}

int nck(int n, int k) {
        return factorial(n) / factorial(k) / factorial(n - k);
}
```

```
void nck_precompute(int** arr, int n) {
        for (int i = 0; i < n; i++)
                arr[i][0] = arr[i][i] = 1;

        for (int i = 1; i < n; i++)
                for (int j = 1; j < i; j++)
                        arr[i][j] = arr[i - 1][j - 1] + arr[i - 1][j];
}
```

```
int nck(int n, int k) {
	if (n - k < k)
		k = n - k;

	int res = 1;
	for (int i = 1; i <= k; i++)
		res = res * (n - (k - i)) / i;

	return res;
}
```