Randomized algorithms

Algorithms that uses **random numbers.**

Las Vegas algorithms.

- The answer is always correct
- Random numbers impact running time $\Rightarrow$ **expected running time**

Monte Carlo algorithms.

- The running time is not affected
- Sometimes give incorrect answer $\Rightarrow$ **probability of error** p
  - One-sided errors (e.g. "no" always correct)
  - Two-sided errors

Important: Running time / probability of error **does not depend on the input instance**, only on choice of random numbers! (i.e. no "systematically bad" input)

Kruskal algorithm (1956)

```
MST-KRUSKAL(E):
  repeat:
    (u,v) := shortest edge
    T := T + {(u,v)}
    contract edge (u,v)
```

Running time: $O(m \log n)$
(use UNION/FIND-SET data structure)

Prim algorithm (1957)

```
MST-PRIM(E):
  s := any starting vertex
  repeat
    (s,v) := shortest edge from s
    T := T + {(s,v)}
    contract edge (s,v)
```

Running time: $O(m \log n)$
with Fibonacci heaps: $O(n \log n + m)$

Borůvka algorithm (1926)

```
MST-BORUVKA(E):
  repeat
    for each vertex v[i]:
       e[i] := shortest edge from v[i]
    T := T + {e[1],e[2],...}
    contract edges e[1],e[2],...
```

Running time: $O(m \log n)$
(in each step, we remove half of vertices)

Randomized algoritmus (Karger et al. 1994)

```
MST-RANDOMIZED(E):
  repeat
1:   2x Borůvka step
2:   R := random subgraph (keep edge with probability p)
3:   F := MST-RANDOMIZED(R)
4:   H := heavy edges with respect to F
5:   E := E - H
```

```
F:={}; R:={}
for each edge e from shortest to longest:
  if e does not introduce cycle to F: MARK e
  with probability p:
     R := R + {e}
     if e is MARKED: F := F + {e}
```

R is a random graph from step 2
F is minimum spanning tree of R
Only MARKED edges can be light