**Lecture Outline (Week 13)**
**Topics in Graph Algorithms (CSCI5320-20S)**

CAI Leizhen
Department of Computer Science and Engineering
The Chinese University of Hong Kong
lcai@cse.cuhk.edu.hk

April 29, 2020

**Keywords:** Randomized algorithms: random selection and random orientation.

# 1 Random Selection

The basic idea is to randomly select an element for a solution with probability at least $p(k)$.

**Example 1**: Feedback Vertex Set revisited: $O(4^k kn)$ randomized algorithm.

*Determine whether a graph $G = (V, E)$ contains at most $k$ vertices $V'$ whose deletion results in an acyclic graph, i.e., a forest.*

**Lemma 1.1** [Voss 1968] *For any feedback vertex set $V'$ in a graph $G$ of minimum degree at least 3, $G - V'$ contains less than half of the edges of $G$.*

**Proof.** Since $G - V'$ is a forest, $e(G - V') < |V - V'|$ where $e(G - V')$ denotes the number of edges in $G - V'$. For vertices in $V - V'$, we have

$$\sum_{v \in V - V'} d(v) = 2e(G - V') + |[V', V - V']|,$$

where $|[V', V - V']|$ is the number of edges in the cut $[V', V - V']$, and

$$\sum_{v \in V - V'} d(v) \geq 3|V - V'|$$

as the minimum degree of $G$ is at least 3. It follows that $|[V, V - V']| > e(G - V')$, and hence $G - V'$ contains less than half of the edges of $G$. ∎

**Implication of Lemma 1.1**: A randomly selected edge $e$ has probability at least $\frac{1}{2}$ to not reside in $G - V'$, implying that at least one end of $e$ belongs to $V'$

and thus a randomly chosen end of $e$ has probability at least $\frac{1}{4}$ to be in solution $V'$.

Becker, Bar-Yehuda and Geiger (1999) observed that we can use Lemma 1.1, together with the following two reduction rules to reduce the problem to graphs of minimum degree at least 3, to derive a randomized $O(kn)$-time algorithm that finds a $k$-vertex feedback vertex cover with probability at least $4^{-k}$: *Randomly choose an edge $e$, randomly put one of its ends $v$ into $V'$, and delete $v$ from $G$.*

**Rule 1.** If vertex $v$ has degree 1, then delete it.

**Rule 2.** If vertex $v$ is adjacent to exactly two vertices $x$ and $y$, then delete it and add edge $xy$.

**Example 2**: Long Path (Gabow and Nie 2008)

*Find an $(s,t)$-path of length at least $k$ in a graph $G$.*

To obtain an FPT algorithm, we can fix a vertex $x$ (only $n$ different choices for $x$) and then find an $(s,x)$-path and an $(x,t)$-path of length exactly $k$. Let $P$ be an $(s,t)$-path in $G$ of length at least $k$, and call the last $k$ vertices in $P$ the *tail* $T$ of $P$. We will use the idea of random selection to guess vertices in $T$.

First we compute an $(s,x)$-path $Q$ in $G$. If $G - V(Q)$ contains an $(x,t)$-path $P'$ that contains $T$ and has length $k$, then $QP'$ is a solution. Otherwise $Q$ must contain some vertices of $T$.

If $Q$ has $\leq k$ vertices, then we just randomly put a vertex of $Q$ into $T$. Otherwise, $Q$ contains more than $k$ vertices, and we denote the $k$-th vertex of $Q$ by $v$. If no vertex of $T$ resides in $Q[s,v]$, then $G$ has a $(v,t)$-path $P^*$ vertex-disjoint from $Q[s,v]$, and $Q[s,v]P^*$ forms a solution. Otherwise, some vertex of $T$ resides in $Q[s,v]$ and we randomly choose a vertex from $Q[s,v]$ and put it into $T$. The correctness of each guess is at least $1/k$, and we can use the color coding technique to find an $(x,t)$-path of length $k$ that contains $T$.

## 2 Random orientation

We randomly orient each edge of a graph to reduce a problem on a graph to some problem on a digraph.

**Example 1**: $k$-Path (Alon, Yuster and Zwick 1995)

Choose a random permutation $\pi : V \to \{1,\ldots,n\}$ of vertices of $G$ and orient an edge $uv$ from $u$ to $v$ iff $\pi(u) < \pi(v)$ to obtain a dag $\vec{G}$.

A $k$-path has probability $\frac{2}{k!}$ to become a directed $k$-path in $\vec{G}$. Since a longest path in a dag can be found in $O(m+n)$ time, we can determine whether $\vec{G}$ has

a directed $k$-path in $O(m+n)$ time, and thus determine whether $G$ has a $k$-path in $O(k!(m+n))$ expected time.

**Example 2**: Partial Vertex Cover (Cai 2016.4.12)

*Find fewest vertices in a graph $G$ to cover at least $k$ edges.*

We may assume $\Delta(G) < k$. Randomly orient each edge to obtain a digraph $\vec{G}$ from $G$, and call a solution $X$ *well-oriented* if all edges in the cut $[X, V-X]$ are oriented from $X$ to $V-X$.

Since $\Delta(G) < k$, $G$ always has a solution covering between $k$ and $2k-2$ edges. It follows that for a random orientation, there is a solution with probability at least $2^{-(2k-2)}$ to become a well-oriented solution.

When $\vec{G}$ contains a well-oriented solution, we can find it in linear time: order vertices nonincreasingly by their out-degrees, and choose vertices until the sum of their out-degrees is at least $k$.

We can improve the probability of a well-oriented solution from $4^{-(k-1)}$ to $2^{-k}$ by changing a well-oriented solution to the following: at least $k - |E(G[X])|$ edges in $[X, V-X]$ are oriented from $X$ to $V-X$.