Lecture 3 (19.10.2012) Author: Mateusz Baranowski

Colour Coding

1 Randomized approach

k-Path
Input: A graph G (directed or not) and a positive integer k
Question: Simple path P of length k

The k-Path algorithm is from the paper that introduced colour coding (Alon, Yuster, Zwick, 1994 [3]).

Let us note that if G is a DAG then the k-Path problem can be easily solved in $O(|V(G)| + |E(G)|)$ using topological sort and a simple dynamic algorithm, whereas for arbitrary graphs the k-Path problem is NP-hard. This can be easily seen by observing that a cubic graph contains a Hamiltonian cycle if and only if it can be made Eulerian by edge deletions.

Let us consider the following algorithm. We number the vertices of G at random with unique values from 1 to $|V|$ and consider a DAG G' such that $V(G') = V(G)$ and $(v_i, v_j) \in G'$ iff $(v_i, v_j) \in G \wedge i < j$. Now we search for a k-Path in G'. As stated earlier this problem can be solved in polynomial time. If a k-path exists in G' then it exists also in G.

Let us assume that P is a k-path in G. We note that $\mathbb{P}[\text{nodes of } P \text{ are in order}] \geq \frac{2}{k!}$. Thus repeating the selection $\frac{k!}{2}$ times we get $(1 - \frac{2}{k!})^{\frac{k!}{2}} \approx \frac{1}{e}$.

We get a randomized algorithm (with probability $\frac{1}{e}$) working in $\mathcal{O}^\star(k!)$.

Theorem 1. *k-Path can be solved in $\mathcal{O}^\star((2e)^k)$ time with probability $\frac{1}{e}$.*

Proof. Let us select a random vertex colouring $c : V \to \{1, 2, \ldots, k\}$. We will be searching for a path which contains all k colours. We will achieve this using dynamic programming, where a subproblem is of the form: "is there a path ending at v which has all the colours in A", which we will represent as a `Boolean` array $T[v, A]$, where $v \in V(G)$ and $A \subseteq \{1, 2, \ldots, k\}$.

function kPath_colour(G, k)
  $\forall_{v \in V} \forall_{A \subseteq \{1,\ldots,k\}} T[v, A] \leftarrow 0$.
  for $v \in V$ **do**
    $T[v, \{c(v)\}] \leftarrow 1$
  end for
  $i \leftarrow 1$
  for $i < k$ **do**
    for $T[u, A] = 1 \wedge |A| = i$ **do**
      for $w \in \mathcal{N}(u)$ **do**
        if $c(w) \notin A$ **then**
          $T[w, A \cup \{c(w)\}] \leftarrow 1$
        end if
      end for
    end for
    $i \leftarrow i + 1$

end for
end function

Using the calculated array we can easily retrieve the k-path or state that it doesn't exist. As the algorithm is only a series of iterations over one set of size $O(2^k)$ and a series of polynomial sets we get the time complexity $\mathcal{O}^\star(2^k)$.

Let P be a k-path in G. What is the probability that P has k colours? At the beginning the starting node can be coloured by any of the k colours. For the second node we can use only $k-1$ colours (we cannot use the colour of the first node). The third node has $k-2$ possible colours and, so on.

We get $k!$ good colourings of a total k^k colourings, so the probability is $\frac{k!}{k^k}$. Using the Stirling approximation

$$k! = \sqrt{2\pi k}(\frac{k}{e})^k(1+o(1))$$

we get $\frac{k!}{k^k} = \Theta(\frac{\sqrt{k}}{e^k})$ Thus by repeating this algorithm e^k times we get probability of success equal to $\frac{1}{e}$ and time complexity $\mathcal{O}^\star((2e)^k)$. □

Eulerian Edge Deletion
Input: An undirected graph G
Question: Delete at most k edges from G so that G becomes Eulerian.

The Eulerian Edge Deletion example is from [6].

Definition 2. A graph $G = (V, E)$ is Eulerian iff

1. $\forall_{v \in V} 2 | \deg(v)$
2. G is connected

Definition 3. Let $G = (V, E)$ and $T \subseteq V$. We call $F \subseteq E$ a T-JOIN if in the graph (V, F) T is the set of vertices with an odd degree.

Theorem 4. MINIMUM T-JOIN $\in \mathcal{P}$

Proof. $\forall_{u,v \in T}$ we calculate $\text{dist}(u, v)$. Now we find the minimal weighted matching for the graph $(T, \{(u, v, \text{dist}(u, v)) : u, v \in T\})$ in polynomial time. This matching is the MINIMUM T-JOIN. □

We see that we can easily remove the minimal number of edges so that all vertices have an even degree. A more complex task is to ensure that this set does not disconnect the graph. For this purpose we will try to establish a witness for connectivity, which will be a spanning tree.

Definition 5. Let $G = (V, E)$ be a graph. If S is a solution to the EED problem, then W is a *connectivity witness* if $E(S) \cap E(W) = \emptyset$ and $\forall_{u,v \in V(S \cup W)} \exists_{P \subseteq W} P$ is a path from u to v.

Definition 6. We will call a vertex v a *terminal* if $2 \nmid \deg(v)$.

Definition 7. An edge $uv \in E$ is called *close* if $\text{dist}(u, T) \leq k$ or $\text{dist}(v, T) \leq k$. Otherwise it is *far*.

Lemma 8. *For every solution S (of* EED*) there exists a connectivity witness W which has $O(k^2)$ close edges.*

Proof. Let $D = V(S)$. We have $|D| \leq 2k$ since S has at most k edges. Let us take a witness W which has exactly i connected components and $(|D| - i)(2k + 2)$ close edges and i is minimal (such i and W always exists as we can take $i = |D|$ and $W = (D, \emptyset)$). If $i = 1$ then the lemma holds, so let us assume otherwise. Let P be the shortest path in $G \setminus S$ connecting two different components of W. Denote these components as K_1 and K_2.

We claim that only the first $k+1$ and the last $k+1$ edges of P can be close. If this is true then this contradicts the choice of W.

Suppose that $e \in P$ is close, but is not in the first or last $k + 1$ edges of P. As e is close, it has an endpoint v such that there is a path P', of length at most k, connecting v and T. K_1 and K_2 are different components, thus $P|v \cup P'$ or $v|P \cup P'$ is a shorter path connecting two components of W. This contradicts the choice of P, thus the lemma holds. Note that we actually proved that the number of closed edges is $\leq (2k - 1)(2k + 2)$. □

Let us consider the following algorithm:

1. Mark each close edge with probability $\frac{1}{2}$ as *blue*.
2. Mark the rest of the edges as *green*.
3. Let W_0 be the connected component of the *green* subgraph containing terminals. If there is more than one such component return NO.
4. We find the MINIMUM T-JOIN in the graph induced by the blue edges with both endpoints in W_0. If the solution has not more than k edges we return it, otherwise we return NO

If the instance of the problem had a solution then we will find it with probability at least $2^{-O(k^2)-k}$. Thus we can solve it with constant probability in $2^{-O(k^2)-k}$.

Note that if the probability of colouring a close edge blue is $1 - \frac{1}{k^2}$ then the probability that S is blue is $2^{-O(k \log k)}$ and the probability that W is green is constant. That way we can solve this problem in $2^{O(k \log k)}$ with a constant probability.

Open problem 1 (stated in [6])**.** Can the Eulerian Edge Deletion problem be solved in $\mathcal{O}^\star(c^k)$ time for some constant c?

Open problem 2 (stated in [5])**.** Consider the following problem: we are given a directed graph G and an integer k, and we are to delete at most k edges from G so that each strongly connected component of G is Eulerian. Is this problem FPT, parametrised by k?

Open problem 3 (stated in [4])**.** Consider the following problem: given graphs G and H, count the number of injective homomorphism from H to G. Can this problem be solved in $\mathcal{O}^\star(c^n)$ for some constant c and $n = |V(G)|$?

Open problem 4. In the Feedback Arc Set in Tournaments problem we are given a tournament (directed complete graph) and we ask to reverse a minimum number of arcs to make it acyclic. Can this problem be solved in $2^{o(n)}$ time?

The $\mathcal{O}^\star(2^{o(k)})$ algorithm for Feedback Arc Set in Tournaments is from [2].

2 Derandomization

2.1 Splitters

Let $[n] = \{1, 2, \ldots, n\}$.

Definition 9. A (n, k, ℓ)-splitter is a family $\mathcal{F}$ of functions $[n] \to [\ell]$, such that for any $W \subseteq [n]$ of size k there exists $f \in \mathcal{F}$ that is injective on W.

What we need is a small family of (n, k, k)-splitters. The following bounds are now known:

Theorem 10 ([8])**.** *There exists a (n, k, k)-splitter of size $e^k k^{\mathcal{O}(\log k)} \log n$ that can be constructed in time $\mathcal{O}(e^k k^{\mathcal{O}(\log k)} n \log n)$.*

Theorem 11 ([8])**.** *There exists a (n, k, k^2)-splitter of size $\mathcal{O}(k^6 \log k \log n)$ that can be constructed in time $\mathcal{O}(k^{\mathcal{O}(1)} n \log n)$.*

On this lecture we prove the following results of [7, 9].

Theorem 12. *There exists a (n, k, k^2)-splitter of size $\mathcal{O}(k^4 \log^2 n / \log\log n)$ that can be constructed in polynomial time.*

Theorem 13. *There exists a $(n, k, 7k)$-splitter of size $\mathcal{O}(2^{\mathcal{O}(k)} \log^2 n / \log\log n)$ that can be constructed in $\mathcal{O}^\star(2^{\mathcal{O}(k)})$ time.*

Let $p > n$ be a prime; we know that p is not much bigger than n. It is not hard to prove that $p < 2n$ and in fact p is typically of order $n + \Theta(\log n)$ due to the prime number theorem:

Theorem 14 (Prime number theorem)**.** *Let $\mathcal{P}(x)$ denote the set of primes not larger than x. Then*

$$\lim_{x \to \infty} \frac{|\mathcal{P}(x)|}{x / \ln x} = 1.$$

Let us fix an integer s. We consider the following function for $1 \le a < p$:

$$f_a(x) = (ax \mod p) \mod s.$$

It can be viewed as a hashing function into s buckets. Consider a set $Z \subseteq [n]$ of k elements we want to hash and let $b_{a,r}$ be a number of elements f_a puts into bucket r, i.e.,

$$b_{a,r} = |f_a^{-1}(r) \cap Z|.$$

The following bound on the number of collisions is crucial:

Lemma 15.

$$\sum_{1 \le a < p} \sum_{0 \le r < s} \binom{b_{a,r}}{2} < \frac{(p-1)k^2}{s}.$$

Proof. Note that the above expression counts the number of triples (x, y, a), $x, y \in Z$, $x \ne y$, $1 \le a < r$, such that f_a collides x and y.

Consider a pair $x, y \in Z$, $x \ne y$. For how many numbers a the function f_a will collide x and y? Well, we need to have $(ax \mod p) \mod s = (ay \mod p) \mod s$. A necessary condition for this is that $a(x-y) \mod p$ is either of the form is or $p - is$ for some $1 \le i < p/s$. As $\mathbb{Z}_p^*$ is a group, there exists exactly one such a for each value is or $p - is$, and we have at most $\frac{2(p-1)}{s}$ such numbers a. As there are $\binom{k}{2} < k^2/2$ pairs (x, y), the lemma follows. □

As a corollary, we obtain that there exists $1 \leq a < p$ such that

$$\sum_{0 \leq r < s} \binom{b_{a,r}}{2} < \frac{k^2}{s}.$$

Note that for $s = k^2$ we obtain $b_{a,r} \leq 1$ for all r. Thus, taking f_a for all $1 \leq a < p$ and $s = k^2$, we obtain a (n, k, k^2)-splitter of size $n + o(n)$.

A simple trick will now decrease the dependency on n to $\log^2 n$. We need the following corollary of the prime number theorem.

Lemma 16. *Recall that $\mathcal{P}(x)$ is the set of prime numbers not larger than x. Then*

$$\ln \prod_{p \in \mathcal{P}(x)} p = x \pm o(x).$$

A note on the margin: an upper bound a bit weaker than the statement of Lemma 16 can be easily proved.

Lemma 17. *For any natural n, the product of all primes not greater than n is at most 4^n.*

Proof. Note that a prime $n < p \leq 2n$ divides $\binom{2n}{n}$, thus a product of all primes between $n/2$ and n is at most 2^n. □

However, in the next lemma we will need the lower bound on the product of primes.

Lemma 18. *Assume $x_1, x_2, \ldots, x_k$ are distinct positive integers not larger than n. Then there exists a prime $q = \mathcal{O}(k^2 \log n)$ such that $x_i \mod q$ are pairwise distinct and non-zero.*

Proof. Let

$$M = \prod_{i=1}^{k} x_i \prod_{1 \leq i < j \leq k} |x_i - x_j|.$$

Note that

$$M \leq n^{k + \binom{k}{2}} \leq n^{k^2}.$$

By Lemma 16, M is smaller than the product of all primes up to $\mathcal{O}(k^2 \log n)$. Therefore there exists a prime $q \leq \mathcal{O}(k^2 \log n)$ that does not divide M, and this prime satisfies the conditions of the lemma. □

Proof of Theorem 12. We pipeline Lemma 18 with the splitter obtained from Lemma 15. That is, for each prime $q \leq k^2 \log n$ and for each integer $1 \leq a < q$, we take to the splitter a function

$$f_{q,a} = (ax \mod q) \mod k^2.$$

□

To prove Theorem 13, we will further hash the universe $[k^2]$. That is, we show a $2^{\mathcal{O}(k)}$-size $(k^2, k, 7k)$-splitter.

We need the following application of Lemma 15. Now $Z \subseteq [k^2]$, $|Z| = k$, and $p > k^2$ is a prime $(p = k^2 + o(k^2))$.

Lemma 19. *For $s = k$ there exists $1 \leq a < p$ such that*

$$\sum_{r=0}^{s-1} b_{a,r}^2 < 3k.$$

Proof. Note that $\sum_{r=0}^{s-1} b_{a,r} = k$ and there exists a such that

$$\sum_{r=0}^{s-1} \binom{b_{a,r}}{2} < k.$$

□

The idea is now as follows. We do not know the set Z. We:

1. branch into $\mathcal{O}(k^2)$ subcases, guessing the good number a;
2. branch into at most $\binom{4k}{k}$ subcases, guessing the values $b_{a,r}$;
3. for each bucket r with $b_{a,r} > 1$, we hash it again with the same prime p and $2b_{a,r}^2$ buckets.

In the end we get at most $k + 2 \cdot 3k = 7k$ buckets.

The crux now is to do a limited guessing in the last step. By Lemma 15, for $s = 2b_{a,r}^2$, at least half of the values $1 \leq a' < p$ do not cause any collisions. Thus, there exists a choice of a' that is good for at least half of the buckets (a random one is such a choice, but we need only existence here). Thus, we guess this a', guess the buckets that are happy with this value a', and forget about them. We spend $\mathcal{O}(k^2 \cdot 2^k)$ time and decrease the number of buckets with collisions by half. Thus, in the end, we get $k^{\mathcal{O}(\log k)} 4^k$ branches.

This concludes the proof of Theorem 13.

2.2 Splitters in action

Let us start with the derandomization of the k-Path algorithm. We will use the $(|V|, k, k)$-splitters of size $e^k k^{O(\log k)} \log |V|$. Instead of colouring at random we check all the functions (k colourings) in the splitter. Let us assume that there exists a path $P \subseteq G$ where $|P| = k$. Since for each subset of $|V|$ of size k there exists an injective function in the splitter then there also exists one for P, but $|P| = k$. Thus one of the functions in the splitter colours P with k different colours.

The running time for a single colour is $\mathcal{O}^\star(2^k)$ and we have $\mathcal{O}^\star(e^k k^{O(\log k)})$ colourings in the splitter so the time complexity is $\mathcal{O}^\star((2e)^k k^{O(\log k)})$.

Now, we will use the (n, r, r^2)-splitters of size $O(r^6 \log r \log n)$ to show how to derandomize the EED algorithm. n will be the number of close edges and $r = (2k-1)(2k+2) + k$, which is the maximum number of close edges a close edge minimal witness can contain. Instead of selecting a random colouring we will try every function f in a (n, r, r^2)-splitter. For this function we will try every set $F \subset \{1, 2, \ldots, r^2\}$ of size k. If e is a close edge then we will colour it blue iff $f(e) \in F$.

If a solution S with a witness W exist then, by the definition of the splitter, there exists a function f that is injective on the set of close edges in $S \cup W$. Furthermore, there exists a set $F \subset \{1, 2, \ldots, r^2\}$ of size k with $f^{-1}(F) \cap W = \emptyset$ and $S \subseteq f^{-1}(F)$. Thus, if a solution exists then

using one of the above colouring functions we will colour a witness green ensuring that it will not be removed during the MINIMAL T-JOIN stage of the algorithm. Note that the number of subsets $F = 2^{O(k \log k)}$ and the size of the selected splitter is polynomial, thus the derandomized algorithm is also FPT.

References

[1] Susanne Albers, Alberto Marchetti-Spaccamela, Yossi Matias, Sotiris E. Nikoletseas, and Wolfgang Thomas, editors. *Automata, Languages and Programming, 36th International Colloquium, ICALP 2009, Rhodes, Greece, July 5-12, 2009, Proceedings, Part I*, volume 5555 of *Lecture Notes in Computer Science*. Springer, 2009.

[2] Noga Alon, Daniel Lokshtanov, and Saket Saurabh. Fast FAST. In Albers et al. [1], pages 49–58.

[3] Noga Alon, Raphael Yuster, and Uri Zwick. Color-coding. *J. ACM*, 42(4):844–856, 1995.

[4] Omid Amini, Fedor V. Fomin, and Saket Saurabh. Counting subgraphs via homomorphisms. In Albers et al. [1], pages 71–82.

[5] Katarína Cechlárová and Ildikó Schlotter. Computing the deficiency of housing markets with duplicate houses. In Venkatesh Raman and Saket Saurabh, editors, *IPEC*, volume 6478 of *Lecture Notes in Computer Science*, pages 72–83. Springer, 2010.

[6] Marek Cygan, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Ildikó Schlotter. Parameterized complexity of eulerian deletion problems. In Petr Kolman and Jan Kratochvíl, editors, *WG*, volume 6986 of *Lecture Notes in Computer Science*, pages 131–142. Springer, 2011.

[7] Michael L. Fredman, János Komlós, and Endre Szemerédi. Storing a sparse table with 0(1) worst case access time. *J. ACM*, 31(3):538–544, 1984.

[8] Moni Naor, Leonard J. Schulman, and Aravind Srinivasan. Splitters and near-optimal derandomization. In *FOCS*, pages 182–191. IEEE Computer Society, 1995.

[9] Jeanette P. Schmidt and Alan Siegel. The spatial complexity of oblivious k-probe hash functions. *SIAM Journal of Computing*, 19(5):775–786, 1990.