Universal Optimality of Dijkstra ALgorithm

Using Fibonacci-Like Priority Queue with Working Sets

Soham Chatterjee

July 21, 2025

Oral Qualifier, STCS

Table of contents

1. Introduction

2. Lower Bounding Query Complexity

Introduction

Metropolis

The **metropolis** theme is a Beamer theme with minimal visual noise inspired by the HSRM Beamer Theme by Benjamin Weiss.

Enable the theme by loading

```
\documentclass{beamer}
\usetheme{metropolis}
```

Note, that you have to have Mozilla's *Fira Sans* font and XeTeX installed to enjoy this wonderful typography.

Sections

Sections group slides of the same topic

```
    \section{Elements}
```

for which **metropolis** provides a nice progress indicator . . .

Lower Bounding Query Complexity

$OPT_Q(G) = \Omega(\log(\mathrm{Order}(G)))$

Theorem

For any directed or undirected graph G, any algorithm for the DO problem needs $\Omega(\log(\mathrm{Order}(G)))$ comparison queries in expectation.

- Let A is any correct algorithm and $L \in \mathrm{Order}(G)$.
- Given L we have a weight assignment w_L such that L is unique order obtained from w_L upon running Dijkstra. For each L fix w_L. Let $\mathcal{W}$ be the collection of all such w_L.
- Let $C_L \in \{-1, 0, 1\}^*$ be the sequence of answers of comparisons made by A on (G, w_L). Then $\mathcal{C} : \mathcal{W} \to \{-1, 0, 1\}^*$, $\mathcal{C}(w_L) = C_L$ is a ternary prefix free code.
- By Shannon's source coding theorem for symbol codes any such code has expected length $\Omega(\log(|\mathcal{W}|)) = \Omega(\log(\mathrm{Order}(G)))$

Barrier Sequence

Let T be any tree. A *Barrier*, $B \subseteq V(T)$ is a set of nodes where for any two vertices $u, v \in B$, u is not ancestor of v in T.

For two disjoint barriers, $B_1 \prec B_2$ if no node of B_2 is predecessor of a node in B_1.

$(B_1, \ldots, B_k)$ is a *barrier sequence* if whenever $i < j$, $B_i \prec B_j$.

Theorem

A sequence $(B_1, \ldots, B_k)$ of pairwise disjoint vertex sets is barrier sequence if and only if for all $1 \leq i \leq j \leq k$, $v \in B_j$ is not ancestor of any $u \in B_i$ in T.

Barriers Give Lower Bounds

Theorem

Let T be any spanning tree and $(B_1, \ldots, B_k)$ be a barrier sequence of T. Then $\log(\text{Order}(G)) = \Omega\left(\sum_{i=1}^{k} |B_i| \log |B_i|\right)$

- We have $\log(\text{Order}(G)) \geq \log(\text{Order}(T))$. We'll show $\log(\text{Order}(T)) \geq |B_1|!|B_2|! \cdots |B_k|!$.
- Delete vertices of B_k to get T'. By induction for the barrier sequence $(B_1, \ldots, B_{k-1})$ for T', $\log(\text{Order}(T')) \geq |B_1|!|B_2|! \cdots |B_{k-1}|!$.

Barriers Give Lower Bounds

- We can order vertices of B_k in any order we want. There are $|B_k|!$ many orders.
- For each order of B_k and any order of $\text{Order}(T')$ we can just concatenate them to get an order of T.

So finally we got the result:

Result

If T is a spanning tree of G and $(B_1, \dots, B_k)$ is a barrier sequence for T then

$$OPT_Q(G) = \Omega\left(\sum_{i=1}^{k} |B_i| \log |B_i|\right)$$

Construction of Barrier Sequence

Consider running Dijkstra algorithm until some time. Let S is the set of nodes that are in the priority queue.

- Notice that S are the leaves of the partial exploration tree built so far which is a subgraph of final exploration tree.
- Therefore, S is an incomparable set of the final exploration tree.
- S forms a barrier.

Result

At any time of the algorithm the set of elements in the priority queue forms a barrier

Intersecting Coloring

Definition (Intersecting Coloring)

An intersecting coloring of $\mathcal{I}$ with k colors is a function $C : \mathcal{I} \to [k]$ that assigns a color to every interval and additionally for every color $i \in [k]$, $\bigcap\limits_{I \in \mathcal{I}, C(I) = i} I \neq \emptyset$.

Every intersecting coloring induces a barrier sequence in the exploration tree in following way: For any color c,

- $B_c = \{v \in V(G) \mid C(I(v)) = c\}$
- $t_c = \min\{t \mid \forall\ v \in B_c, t \in I(v)\}$
- Order $\{B_c\}$ by increasing order of $\{t_c\}$. WLOG $t_1 < \cdots < t_k$.
- $(B_1, \ldots, B_k)$ is a barrier sequence.

Intersecting Coloring Gives Lower Bounds

Let C be an intersecting coloring of $\mathcal{I}$ with k colors. Let $(B_1, \ldots, B_k)$ is the barrier sequence induced by C. Then let the energy of C is defined to be

$$E(C) = \sum_{i=1}^{k} |B_i| \log |B_i|$$

Result

If $\mathcal{I}$ is the interval set induced by Dijkstra and C be any arbitrary intersecting coloring of $\mathcal{I}$ then

$$OPT_Q(G) = \Omega(E(C))$$

Good Intersecting Coloring gives Optimality

Goal: Find an intersecting coloring of $\mathcal{I}$, C such that $E(C) \geq Cost(\mathcal{I})$

- Then time complexity of all $\textsc{ExtractMin}$ operations is $O(n + Cost(\mathcal{I})) = O(n + E(C))$.
- We have $OPT_Q(G) = \Omega(E(C))$.
- So overall Cost of $\textsc{ExtractMin}$ in Dijkstra is upper bounded by $O(n + OPT_Q(G))$.
- Dijkstra achieves universal optimality for time complexity.

We will find such a good intersecting coloring recursively.

Deleting Intervals from $\mathcal{I}$

Theorem

Let $\mathcal{I}$ an interval set and $x \in \mathcal{I}$. $k = \max_t |\{I \in \mathcal{I} \mid t \in I\}|$. Then

$$Cost(\mathcal{I}) \leq Cost(\mathcal{I} \setminus \{x\}) + \log |W_x| + \log k$$

- Let $I_1, \ldots, I_l \in \mathcal{I}$ are the only intervals which had nonempty intersection with x. So $l \leq k - 1$.
- Let t_i is starting point of I_i. WLOG assume $t_l > \cdots > t_1$.
- Let W_i, W_i' are working sets of I_i before and after removing x.

Deleting Intervals from $\mathcal{I}$

- Let t is starting point of x. Then $W_{i,t}$ contains $x, I_1, \dots, I_i$. So $|W_i| \geq i + 1$.
- $|W_i| \in \{|W_i'|, |W_i'| + 1\}$ for all $i \in [l]$.

$$
\begin{aligned}
& Cost(\mathcal{I}) - Cost(\mathcal{I} \setminus \{x\}) - \log |W_x| \\
= & \sum_{i=1}^{l} \log |W_i| - \log |W_i'| \\
\leq & \sum_{i=1}^{l} \log(i+1) - \log i = \log(l+1) \leq \log k
\end{aligned}
$$

Fact

For any working set $|W_x| = k$ we have

$$Cost(\mathcal{I}) \leq Cost(\mathcal{I} \setminus W_x) + 2|W_x| \log |W_x|$$

Finding Good Intersecting Coloring

We will construct C by induction on $|\mathcal{I}|$.

Questions?

Backup slides

Sometimes, it is useful to add slides at the end of your presentation to refer to during audience questions.

The best way to do this is to include the `appendixnumberbeamer` package in your preamble and call `\appendix` before your backup slides.

metropolis will automatically turn off slide numbering and progress bars for slides in the appendix.

References i