Instead

- Store vertices $v \in V - X$ in heap

- Key of $v$ is $v.\text{key} = \min_{w \in X} \{A[w] + l_{w,v}\}$

- For each element $v_i$, $v_i.\text{p} = \arg\min_{w \in X} \{A[w] + l_{w,v_i}\}$

- If $v$ is not directly connected to $X$, $k(v) = \infty$
  $v.\text{p} = \emptyset$

**Why better?**

$X$ before $w^*$ added

$X$ after $w^*$ added

$\xrightarrow{\text{Only need to check vertices in } A_G[w^*]}$
Dijkstra Heap

X[v] = 0; A[v] = ∞; B[v] = ∅; ∀ v ∈ V
v.key = ∞ for all v ∈ V
v.p = ∅ for all v ∈ V
s.key = 0.

Heapify all v ∈ V

while (Heap is not empty)

- Let w = vertex with min key
- Remove w; X[w] = 1; A[w] = w.key;
- B[w] = B[w.p] + (w.p, w);

- for u ∈ Ag[w] & & u not explored
  - check if need to update u.key
  - if yes, remove and reinsert
**Diagonal BFS Runtime Page 3**

Diagram:

```
  3
 S o   u   s   x
  ↓   1
  v
```

**Example:**

<table>
<thead>
<tr>
<th></th>
<th>Round 1</th>
<th>Round 2</th>
<th>Round 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>H</td>
<td>( S \rightarrow 0</td>
<td>\phi )</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>u</td>
<td>3</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>v</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>x</td>
<td>8</td>
<td>6</td>
</tr>
</tbody>
</table>
Dijkstra Heap

\[ X[v] = 0; \quad A[v] = \infty; \quad B[v] = \emptyset; \quad \forall v \in V \]
\[ v.\text{key} = \infty \quad \text{for all } v \in V \]
\[ v.\text{p} = \emptyset \quad \text{for all } v \in V \]
\[ s.\text{key} = 0. \]

Heapify all \( v \in V \)

\[
\text{while (Heap is not empty)}
\begin{align*}
\quad & \text{Let } w = \text{ vertex with min key} \\
\quad & \text{Remove } w; \quad X[w] = 1; \quad A[w] = w.\text{key} \\
\quad & B[w] = B[w.\text{p}] + (w.\text{p}, w) \\
\quad & \text{for } u \in A_G[w] \& u \text{ not explored} \\
\quad & \quad \text{check if need to update } u.\text{key} \\
\quad & \quad \text{if yes, remove & reinsert}
\end{align*}
\]

How many times does this loop run?

How many times does this check happen over whole algorithm?

What is cost?