Lec 19 Johnson's Algorithm & Graph Contraction

So far:

	Work	Span	Parallelism
Dijkstra	$O(m \lg n)$	$O(m \lg n)$	None
Bellmond-Ford	$O(mn)$	$O(n \lg n)$	$O(\frac{n}{\lg n})$

Those are single source shortest path (SSSP) problems
Asymptotically no better way to find SP given source and target than SSSP

What if we want all pairs shortest path (APSP)?

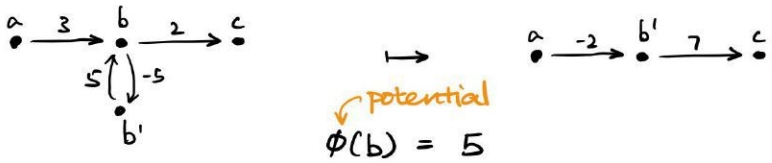
Brute force with Dijkstra?

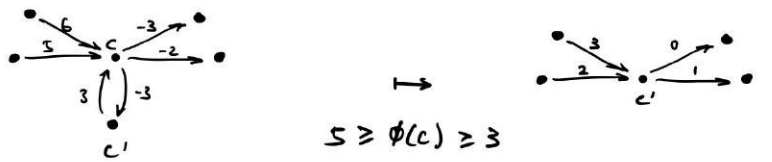
Johnson's Alg

Changing edge weight

Naive: add weight to each edge :(Bad

Potential property:





General case — reweight all edges by potential

$$w(u', v') = w(u, v) + \phi(u) - \phi(v)$$

(φ: potential)

Claim $\delta(u', v') = \delta(u, v) + \phi(u) - \phi(v)$

$(\Leftrightarrow \delta(u, v) = \delta(u', v') - \phi(u) + \phi(v))$ △

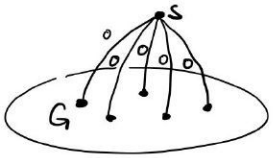
Shortest path relax property

(a, b) relaxed if $\delta(s, a) + w(a, b) \geq \delta(s, b)$

$\Leftrightarrow w(a, b) \geq \delta(s, b) - \delta(s, a)$

$\Leftrightarrow w(a, b) + \delta(s, a) - \delta(s, b) \geq 0$ ← looks like potential

Dummy Vertex



Reduction

1. Pick potentials so that all weights ≥ 0
   i. Add dummy verts s & edges of 0 weight
   ii. Run Bellmond-Ford to get $\delta(s, \cdot)$
   iii. $w(u', v') = w(u, v) + \delta(s, u) - \delta(s, v)$
2. Dijkstra from all source
3. Recover path lengths with △

Cost

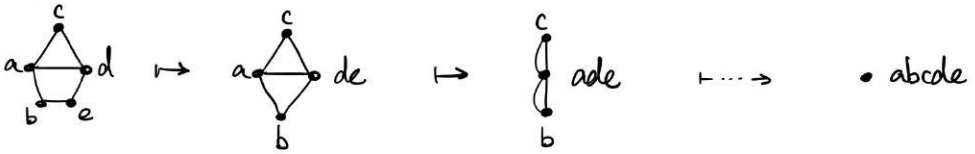
	Bellmond-Ford	Dijkstra	Overall
Work	$O(mn)$	$n\,O(m \lg n)$	$O(mn \lg n)$
Span	$O(n \lg n)$	$O(m \lg n)$	$O(m \lg n)$

Graph contraction — gives you polylog span

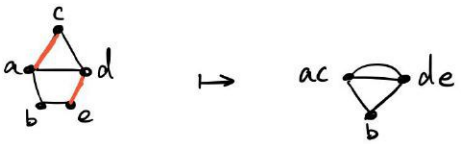
→ Generally search-based algs not good for parallelism. Think line graph causing troubles...

Useful for:
- Graph connectivity
- biconnected components
- Min spanning tree

Edge contraction



Contract by constant factor: find maximal matching and contract



greedy works, but is sequential :c

Parallel mostly maximal matching

Random priority, pick max among touching edges