

A Functional Approach to External Graph Algorithms

Presented By Ryan Tomich

Context behind the paper

- James Abello, Adam L. Buchsbaum, Jeffery R. Westbrook
- AT&T Labs
- Algorithmica in 2002
- Context
 - external-memory model (Aggarwal–Vitter, 1988)
 - Primitives: SCAN, SORT, ...
 - “Fixing” classical graph algorithms for graphs exceeding RAM.

A Functional Approach to External Graph Algorithms¹

J. Abello,² A. L. Buchsbaum,² and J. R. Westbrook³

Abstract. We present a new approach for designing external graph algorithms and use it to design simple, deterministic and randomized external algorithms for computing connected components, minimum spanning forests, bottleneck minimum spanning forests, maximal independent sets (randomized only), and maximal matchings in undirected graphs. Our I/O bounds compete with those of previous approaches. We also introduce a semi-external model, in which the vertex set but not the edge set of a graph fits in main memory. In this model we give an improved connected components algorithm, using new results for external grouping and sorting with duplicates. Unlike previous approaches, ours is purely functional—without side effects—and is thus amenable to standard checkpointing and programming language optimization techniques. This is an important practical consideration for applications that may take hours to run.

Key Words. Connected components, External memory algorithms, Graph algorithms, Maximal matchings, Maximal independent sets, Minimum spanning trees, Functional programming.

1. Introduction. Because classical algorithms often do not scale when data exceed main memory limits, recent attention has focused on algorithms that process data in external storage (disk and tape) [1], [6], [36]. While some external algorithms (e.g., for sorting [2]) closely resemble their RAM analogues, others seem to require different approaches. Graph algorithms for RAMs, in particular, seem poorly suited for direct extension to external memory, because of the lack of locality in graph data. Current approaches for designing external graph algorithms use either PRAM simulation techniques [13], or else external data structures [4], [26] that do not completely address the I/O implications of graph traversal.

We present a new divide-and-conquer approach for designing external graph algorithms, which is simple to describe and implement. No sophisticated data structures are needed. When unwinding the divide-and-conquer recursions, our algorithms effect successions of graph transformations that reduce to sorting, selection, and a recursive bucketing technique. We apply our techniques to devise external algorithms for computing connected components, minimum spanning forests (MSFs), bottleneck minimum spanning forests (BMSFs), maximal independent sets, and maximal matchings in undirected graphs.

We focus on producing algorithms that are purely functional. That is, each algorithm is specified as a sequence of functions applied to input data and producing output data,

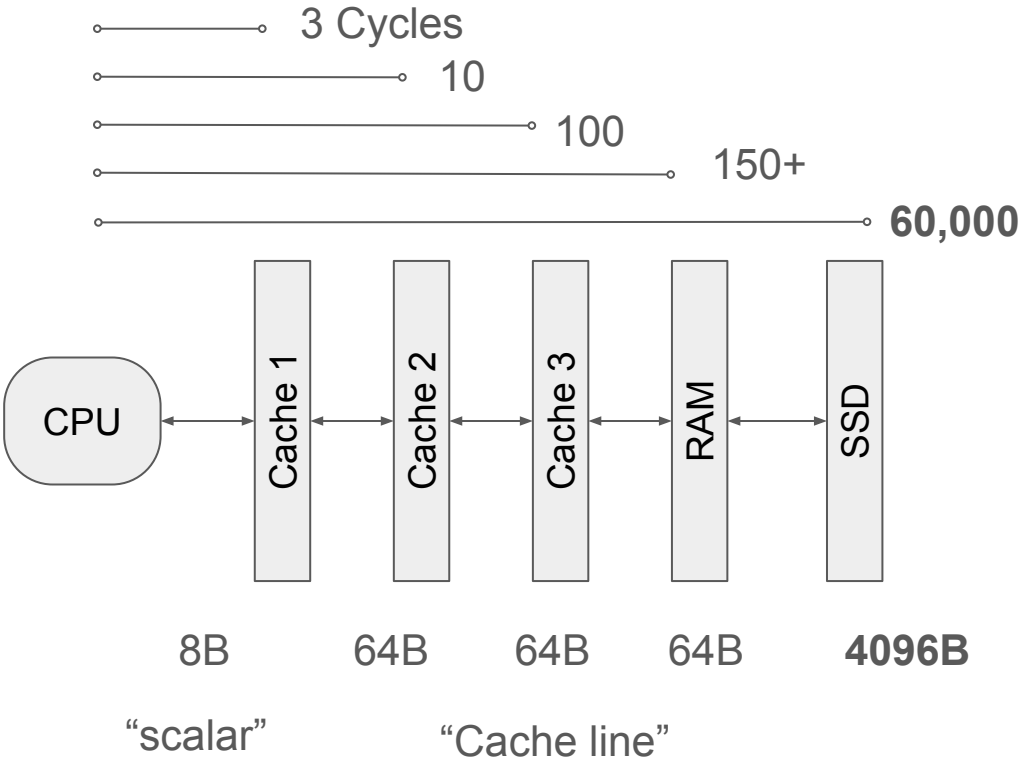
¹ An extended abstract of this work appears in *Proc. 6th European Symposium on Algorithms*, Venice, 1998.

² AT&T Labs, Shannon Laboratory, 180 Park Ave., Florham Park, NJ 07932, USA. {abello, alb}@research.att.com.

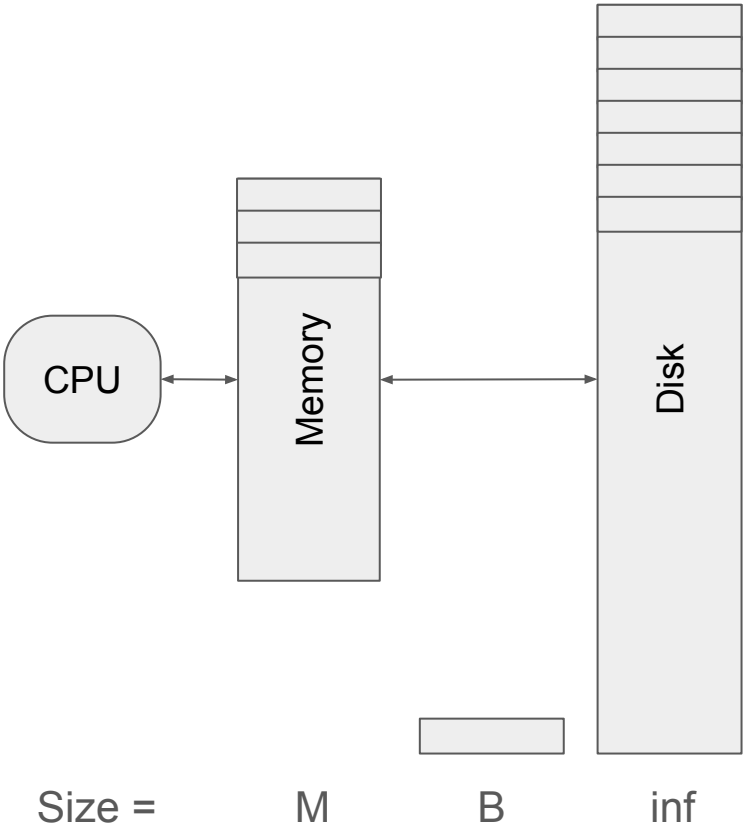
³ 20th Century Television, Los Angeles, CA 90025, USA. jwestbrook@acm.org. Work completed while a member of AT&T Labs.

The External Memory Model

Reality*



I/O Model



I/O Complexity Primitives

SCAN:

- Bring in N elements in groups of size B

$$\text{scan}(N) = \left\lceil \frac{N}{B} \right\rceil$$

SORT

- Merge Sort
- Read and sort M-sized chunks in RAM.
- Merge M/B runs at a time in a buffer

$$\text{sort}(N) = \Theta\left(\frac{N}{B} \log_{M/B} \frac{N}{B}\right)$$

BUCKETING

- Keep M/B bucket buffers in memory
- Scan the input, placing each record into its bucket's buffer
- Write each buffer when full

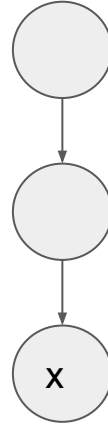
One bucketing pass = $O(\text{scan}(N))$

- Number of buckets k exceed M/B?

$$\Theta\left(\text{scan}(N) \max\left\{1, \log_{M/B} K\right\}\right)$$

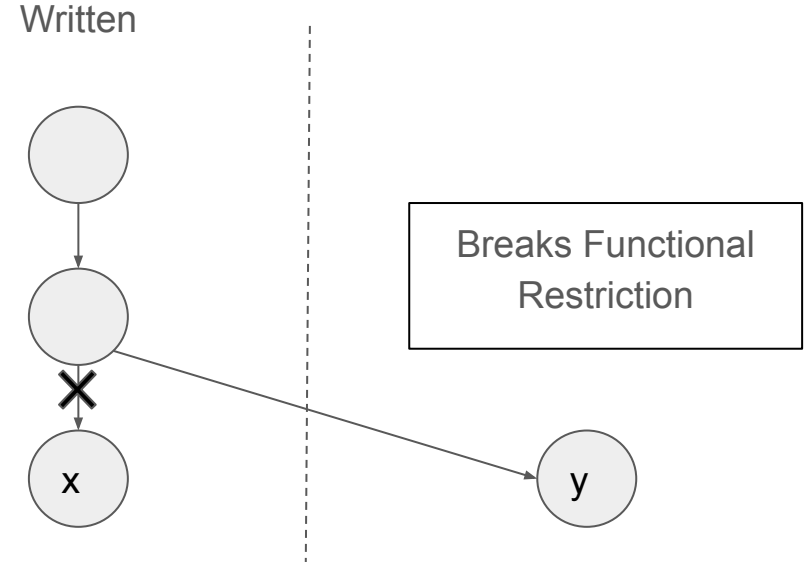
Functional Restriction

- “... information, once written, remains unchanged.”
- Practically, allows for **checkpointing**.



Functional Restriction

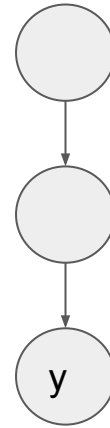
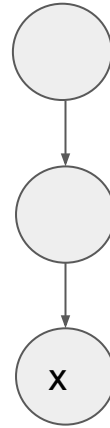
- “... information, once written, remains unchanged.”
- Practically, allows for **checkpointing**.



Functional Restriction

- “... information, once written, remains unchanged.”
- Practically, allows for **checkpointing**.

Written

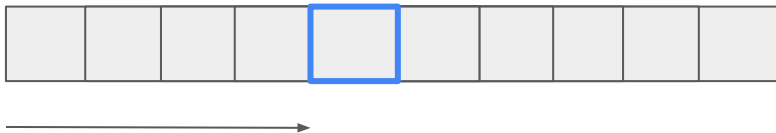


Subroutines

Selection

Find a value of a specified rank

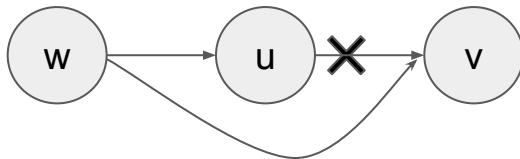
- Median of medians
- Lemma 3.3: $O(\text{scan}(N))$



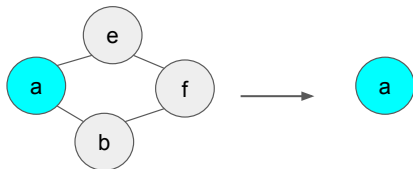
Relabeling

Replace each edge endpoint with its parent

- Sort the mapping (x,y) by x
- Sort edges (u,v) in E by v
- Scan both lists together, creating new edges E' applying the mapping
- Sort edges (u,v) in E' by u
- Scan both lists and replace
- Lemma 3.4: $O(\text{sort}(|E|) + \text{sort}(|V|))$



Contraction



Replace identified groups with super-vertices

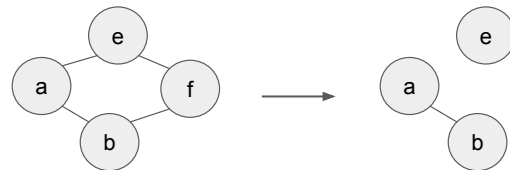
- Scan each supplied group and choose a representative
- Construct a rooted-star mapping
- Use relabeling to apply the mapping to every edge of the graph

Correctness

- Lemma 3.1: This contraction is safe for connected components
- Lemma 3.2: Recursive contractions will correctly compute CC

Lemma 3.5: $O\left(\text{sort}(|I|) + \text{sort}\left(\sum_i |C_i|\right)\right)$

Deletion



Filter out specific edges or vertices

- Sort E, Sort deletion list D
- Scan D and E together and write E only if its absent from D
- Sort deletion list D
- Sort (u,v) in E by u
- scan and discard if d = u
- Repeat for (u,v) in E by v

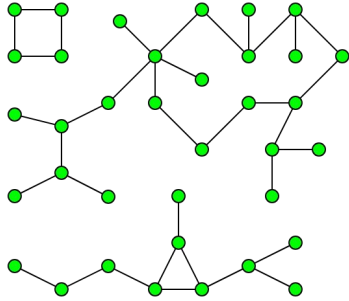
Lemma 3.6: $O(\text{sort}(|E|) + \text{sort}(|D|))$

Algorithms

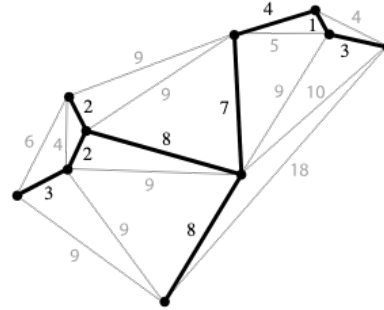
Subroutines: SCAN, SORT, BUCKET,
Selection, Relabeling, Contraction, Deletion

Problems

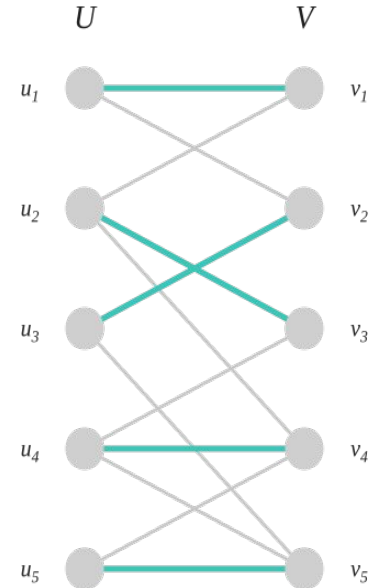
Connected Components



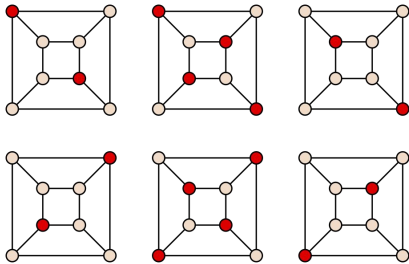
Minimum spanning forests



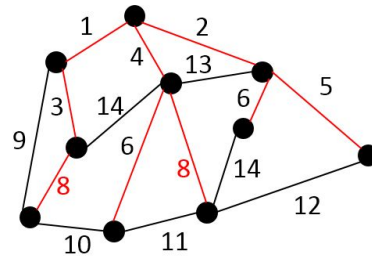
Maximal matching



Maximal independent set



Bottleneck minimum spanning forests



*Images from wikipedia

Deterministic Connected Components

Algorithm CC

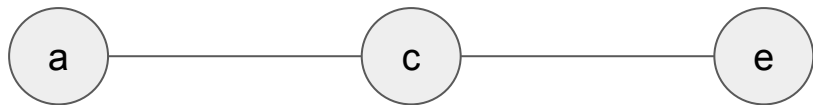
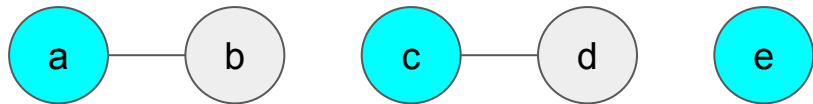
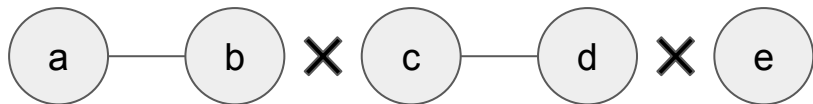
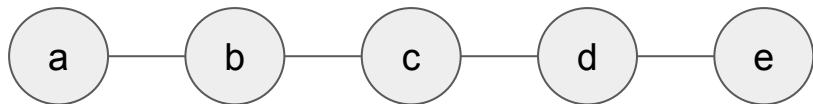
1. Let E_1 be any half of the edges of G ; let $G_1 = (V, E_1)$.
2. Compute $CC(G_1)$ recursively.
3. Let $G' = G/CC(G_1)$. Contraction
4. Compute $CC(G')$ recursively.
5. $CC(G) = CC(G') \cup RL(CC(G'), CC(G_1))$. relabel

Runtime:

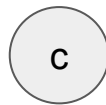
- Choose half the edges $O(\text{scan}(E))$
- Contract in $O(\text{sort}(E))$
- Combine Components $O(\text{Sort}(E))$

$$T(E) \leq 2T(E/2) + O(\text{sort}(E))$$

$$T(E) \text{ in } O(\text{sort}(E)\log(E/M))$$



...



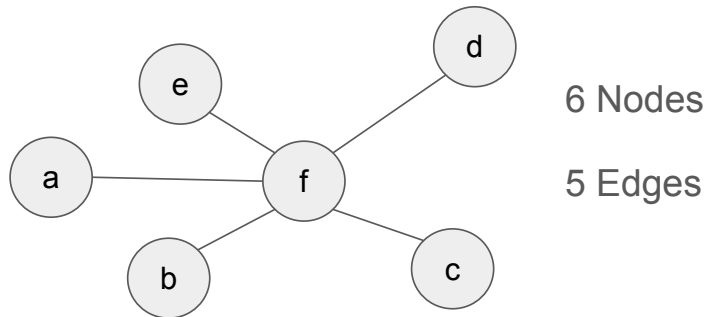
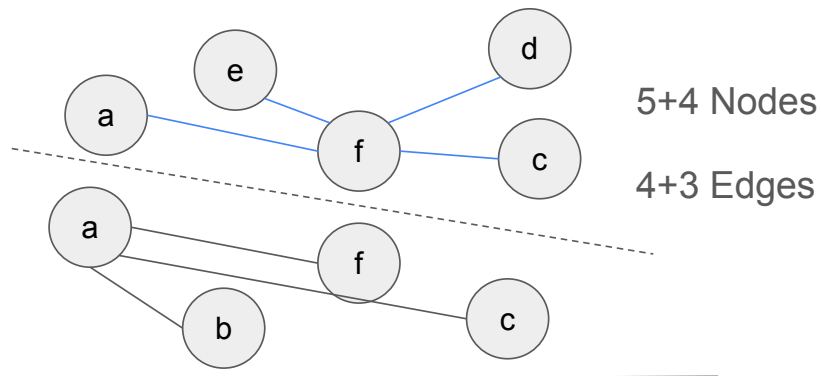
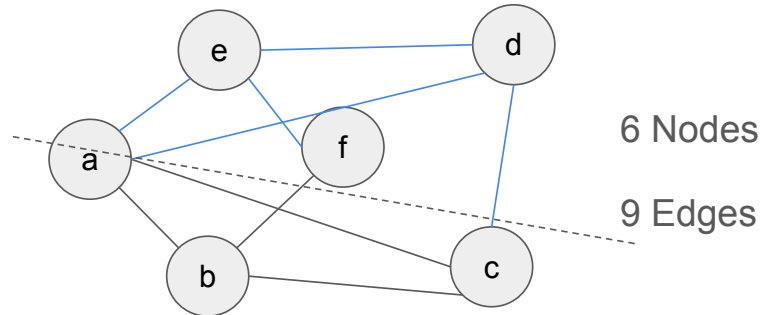
Sparsification

```
SparseCC(G):  
  // summarize both halves, then use the summaries  
  if G fits in RAM:  
    Compute and return its component-star forest  
  
  if G has at most V edges: // E/V groups  
    return BasicCC(G)  
  
  Divide the edge list into two halves G1 and G2  
  F1 = SparseCC(G1)  
  F2 = SparseCC(G2)  
  
  return BasicCC(F1 ∪ F2).
```

Why are they good external algorithms?

- Streaming
- Recursion
- Compression

$$O\left(\text{sort}(E) + \frac{E}{V} \text{sort}(V) \log_2(V/M)\right)$$



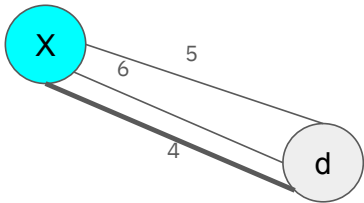
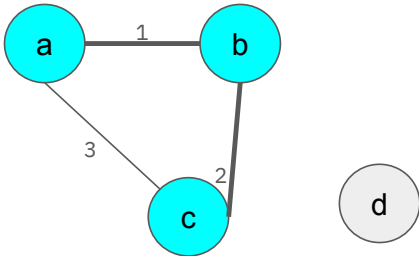
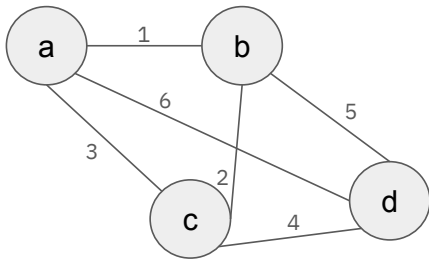
Deterministic Minimum Spanning Forest

Algorithm MSF

1. Let E_1 be any lowest-cost half of the edges of G ; i.e., every edge in $E \setminus E_1$ has weight at least that of the edge of greatest weight in E_1 . Let $G_1 = (V, E_1)$.
2. Compute $MSF(G_1)$ recursively.
3. Let $G' = G / MSF(G_1)$. Contraction
4. Compute $CC(G')$ recursively.
5. $MSF(G) = EX(MSF(G')) \cup MSF(G_1)$. Expand

$$O\left(\text{sort}(E) + \frac{E}{V} \text{sort}(V) \log_2(V/M)\right)$$

Problem	Deterministic
	I/O bound
Connected components	$O(\text{sort}(E) + \frac{E}{V} \text{sort}(V) \log_2 \frac{V}{M})$
MSFs	$O(\text{sort}(E) + \frac{E}{V} \text{sort}(V) \log_2 \frac{V}{M})$
BMSFs	$O(\text{sort}(E) + \frac{E}{V} \text{sort}(V) \log_2 \frac{V}{M})$



Randomized Minimum Span Forest

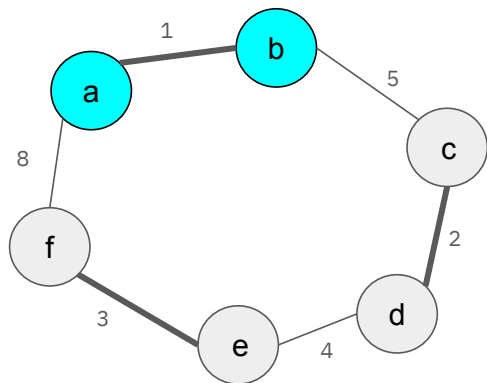
```
RandomizedMSF(G):  
  if G fits in RAM:  
    Compute and return its minimum spanning forest.  
  
  Borůvka x2:  
    vertices select cheapest incident edge  
    Contract the selected trees creating G'  
  
  H = G' keeping each e with prob 1/2  
  
  F1 = RandomizedMSF(H)  
  
  Remove each edge (u,v) from G' that is heavier than  
  the heaviest edge on the path from u to v in F1.  
  Keep edges whose endpoints are not connected in F1.  
  Call the remaining graph G".  
  
  F2 = RandomizedMSF(G")  
  
  Return S ∪ F2, restoring original edge endpoints.
```

Why are they good external algorithms?

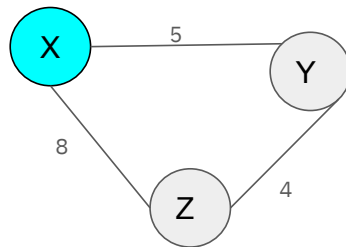
- Streaming
- Recursion
- Compression
- + **Random sampling: subgraph providing evidence to shrink the large graph**

Problem	Randomized	
	I/O Bound	With probability
Connected components	$O(\text{sort}(E))$	$1 - e^{-\Omega(E)}$
MSFs	$O(\text{sort}(E))$	$1 - e^{-\Omega(E)}$
BMSFs	$O(\text{sort}(E))$	$1 - e^{-\Omega(E)}$

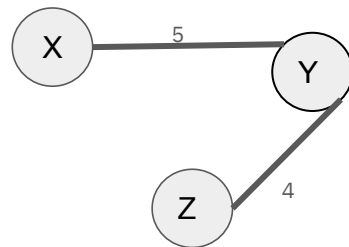
1) Borůvka



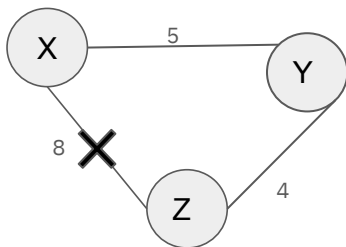
2) Contract their trees



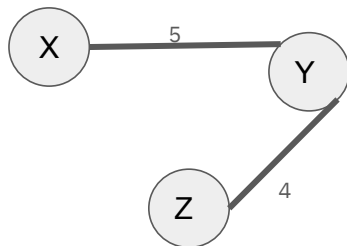
3) Sample Edges and MST



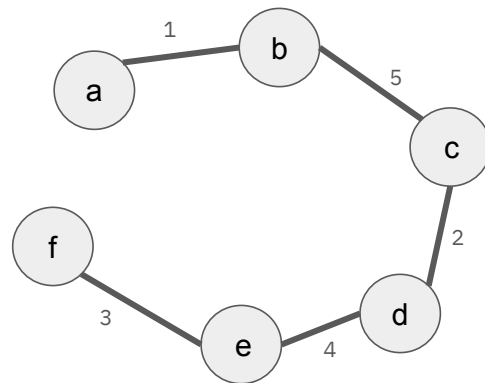
4) Remove large edges



5) Recurse MST



6) Merge



xz_max = 5
8 > w_max (discard)

Generalized Algorithm

```

ALGO(G):
  if G fits in RAM:
    Compute ALGO(G) normally

  Reduce Graph if possible (Borůvka)

  G1 = subgraph with less edges
  P = ALGO(G1)
  Random Sample

  Use P to simplify G to G'
  Contract G'
  Delete using info from random sample

  R = ALGO(G')
  Return Combine(R, P)
    
```

Problem	Deterministic	
	I/O bound	
Connected components	$O(\text{sort}(E) + \frac{E}{V} \text{sort}(V) \log_2 \frac{V}{M})$	
MSFs	$O(\text{sort}(E) + \frac{E}{V} \text{sort}(V) \log_2 \frac{V}{M})$	
BMSFs	$O(\text{sort}(E) + \frac{E}{V} \text{sort}(V) \log_2 \frac{V}{M})$	
Maximal matchings	$O(\frac{E}{V} \text{sort}(V) \log_2 \frac{V}{M})$	
Maximal independent sets		
Problem	Randomized	
	I/O Bound	With probability
Connected components	$O(\text{sort}(E))$	$1 - e^{-\Omega(E)}$
MSFs	$O(\text{sort}(E))$	$1 - e^{-\Omega(E)}$
BMSFs	$O(\text{sort}(E))$	$1 - e^{-\Omega(E)}$
Maximal matchings	$O(\text{sort}(E))$	$1 - \varepsilon$ for any fixed ε
Maximal independent sets	$O(\text{sort}(E))$	$1 - \varepsilon$ for any fixed ε

Semi-External Problems

Nodes fit in RAM, but Edges do not

- CC: Keep union–find in RAM. Scan edges and merge their endpoints' components.
- MSF: Keep forest in RAM. Scan and add edges e ; if e creates a cycle, remove the cycle's heaviest edge.
- MM: Keep each vertex's matched/unmatched status in RAM. Accept an incoming edge if both endpoints are unmatched.

$$O\left(\text{scan}(E) \max\{1, \log_{M/B} C\}\right)$$

Main Takeaways

- IO model
- Subroutines: SCAN, SORT, BUCKET, Selection, Relabeling, Contraction, Deletion
- Sparsification
- Randomization
- Semi-External Problems

```
ALGO(G):  
    if G fits in RAM:  
        Compute ALGO(G) normally  
  
    Reduce Graph if possible (Borůvka)  
  
    G1 = subgraph with less edges  
    P = ALGO(G1)  
    Random Sample  
  
    Use P to simplify G to G'  
    Contract G'  
    Delete using info from random sample  
  
    R = ALGO(G')  
    Return Combine(R, P)
```

Critique

- No experimental validation
- Difficult layers of subroutines that would also need to be fast.
- Contribution and general takeaways were difficult to grasp. Patterns between algorithms was difficult to grasp.
- BMSF is a derived result, overclaimed.

Discussion

- Why do we think these graph problems were picked to showcase External Graphs?

References

Abello, J., Buchsbaum, A. L., and Westbrook, J. R.
A functional approach to external graph algorithms.
Algorithmica, 32:437–458, 2002. doi:
[10.1007/s00453-001-0088-5](https://doi.org/10.1007/s00453-001-0088-5).

Acknowledgement

GPT-6 Astra was used to help the author understand aspects of the algorithms and refine selected examples. The author takes full responsibility for all content.

Jeff Westbrook

3 languages

Article Talk

Read Edit View history

From Wikipedia, the free encyclopedia

Jeff Westbrook is a [TV writer](#) best known for his work on *The Simpsons* and *Futurama*, for which he is a three-time winner of the [WGA Award](#).^{[1][2][3]}

Education and pre-TV [edit]

Prior to becoming a TV writer, Westbrook was a successful [algorithms](#) researcher. After majoring in [physics](#) and [history of science](#) at [Harvard University](#), he studied [computer science](#) with [Robert Tarjan](#) at [Princeton University](#), receiving his [Ph.D.](#) in 1989 with a thesis entitled *Algorithms and Data Structures for Dynamic Graph Algorithms*.^[4] He then took a faculty position at [Yale University](#), later becoming a researcher for [AT&T Laboratories](#) before leaving research for Hollywood.

Erdős and Bacon numbers [edit]

Westbrook's [Erdős number](#) is three due to his research collaborations with Tarjan and others. His [Bacon number](#) is also three, due to his appearance as an extra in the movie *Master and Commander: The Far Side of the World*, giving a combined [Erdős–Bacon number](#) of six.^{[5][6]}

Writing credits [edit]

Futurama episodes [edit]

- "[The Day the Earth Stood Stupid](#)" (2001)
- "[The 30% Iron Chef](#)" (2002)



Jeff Westbrook presenting a talk on the math in *The Simpsons* in the episode "[Treehouse of Horror VI](#)."

Appendix

Maximal Matching

Algorithm **MM**

1. Let E_1 be any non-empty, proper subset of edges of G ; let $G_1 = (V, E_1)$.
2. Compute $MM(G_1)$ recursively.
3. Let $E' = E \setminus V(MM(G_1))$; let $G' = (V, E')$.
4. Compute $MM(G')$ recursively.
5. $MM(G) = MM(G') \cup MM(G_1)$.