CS 341: Algorithms

Lecture 16: Max flow

Éric Schost

based on lecture notes by many other CS341 instructors

David R. Cheriton School of Computer Science, University of Waterloo

Fall 2024

Goals

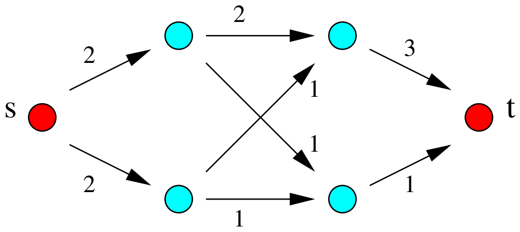
Next three lectures:

- basic results on flows and cuts
- Ford-Fulkerson algorithm for flows
- correctness via max flow = min cut
- some applications

Flows

Setup.

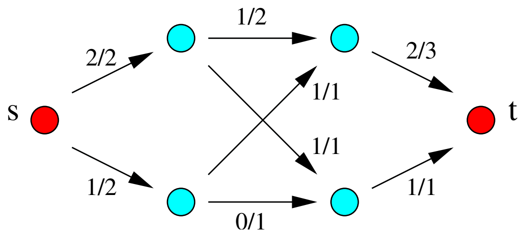
- let G be a **directed** graph, with **no isolated vertex** $(m \geq n/2)$, and let c be a **capacity** on the edges of G
  - for all e, $c(e) \geq 0$
  - by default, $c(e)$ is an **integer**
- we isolate two vertices in G, which will be called the **source** s and the **sink** t. there is no edge going **to** $\boldsymbol{s}$ or **from** $\boldsymbol{t}$.
- we want to send as much "flow" as possible (water in pipes, material on transport networks, ...) while respecting certain rules.



Flows

Definition: a **flow** is a function f of the edges that satisfies

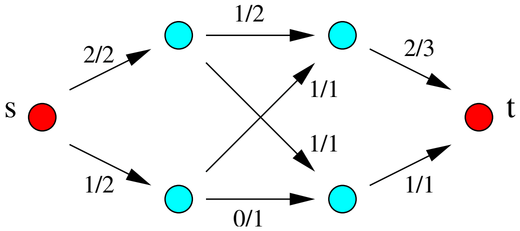
- for any edge e, we have $\mathbf{0} \leq \boldsymbol{f(e)} \leq \boldsymbol{c(e)}$
- the amount of flow that **enters** a vertex equals the amount of flow that **goes out** of it (except at s and t)



Flows

Definition: the **value** of a flow is the amount of flow that goes out of the source:

$$\mathsf{Val}(f) = \sum_{(s,v) \text{ edge}} f(e).$$



here, value is 3.

MaxFlow problem: find a flow with a maximal value.

Producing bananas

Example

We have banana factories $\boldsymbol{F_1, F_2, F_3}$ and grocery stores $\boldsymbol{S_1, S_2}$.

- $\boldsymbol{F_i}$ can produce up to $\boldsymbol{f_i}$ tons of bananas,
- $\boldsymbol{S_j}$ wants $\boldsymbol{s_j}$ tons of bananas.

How to maximize production?

Producing bananas

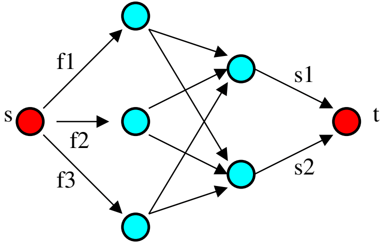
Example

We have banana factories F_1, F_2, F_3 and grocery stores S_1, S_2.

- F_i can produce up to f_i tons of bananas,
- S_j wants s_j tons of bananas.

How to maximize production?

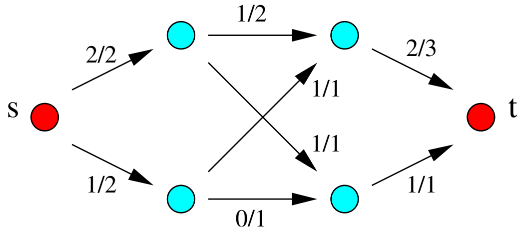
Compute the maximal flow in the following graph (the middle edges have large capacity, such as f_1, f_2, f_3).



Ford-Fulkerson's algorithm

Improving the value

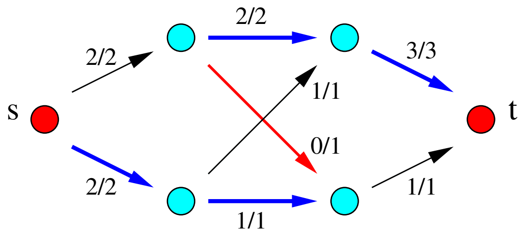
we may have to **decrease** the flow through some edges.



here, we are stuck if we only allow to increase all edges' flow.

Improving the value

we may have to **decrease** the flow through some edges.



we improve the value to 4 by redirecting some flow that was going through the red edge.

amounts to sending one extra flow unit all along the colored path, taking the red edge **backward**.

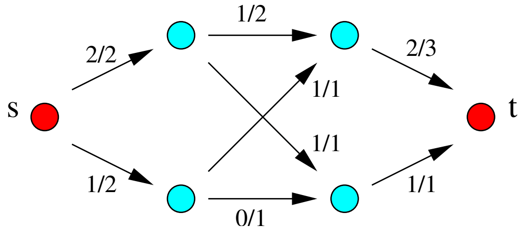
The residual graph

The residual graph G_f shows all the ways to increase the value of the flow.

Definition

- vertices of G_f are those of G.
- for e in E
  - if $f(e) < c(e)$, put e in edge(G_f) with capacity $c(e) - f(e)$
  - if $f(e) > 0$, put **reverse**(e) in edge(G_f) with capacity $f(e)$.

  (edges of G_f show what modifications are possible)



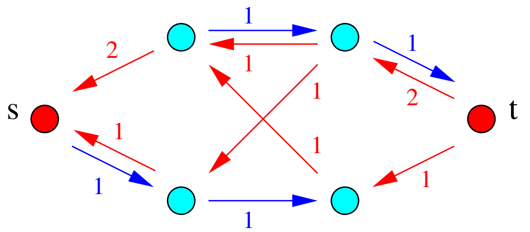
The residual graph

The residual graph G_f shows all the ways to increase the value of the flow.

Definition

- vertices of G_f are those of G.
- for e in E
  - if $f(e) < c(e)$, put e in $\mathsf{edge}(G_f)$ with capacity $c(e) - f(e)$
  - if $f(e) > 0$, put **reverse**(e) in $\mathsf{edge}(G_f)$ with capacity $f(e)$.

(edges of G_f show what modifications are possible)



Using the residual graph

path from s to t in G_f gives a way to increase the value in G.

- **blue edge** of capacity c: can **increase** the flow by up to c on that edge in G
- **red edge** of capacity c: can **decrease** the flow by up to c on the reverse of this edge in G

improvement step:

- compute the residual graph
- find a **(simple) path** γ from s to t in G_f, if one exists, using BFS, DFS, or something else (can assume $\boldsymbol{O(m)}$)
- let $\boldsymbol{x}$ be the **minimal** value of all capacities on γ in G_f
- update the flow on G accordingly

  - increase the blue edges by x

  - decrease the reverse of red edges by x

Correctness of the improvement step

Claim

after an improvement step,

- we still have a flow on G
- the value has increased by x
- if we had an integer flow (and integer capacities in G), still have an integer flow

Proof

- **we still have a flow**
  - all flow values on the edges are ≥ 0 and do not exceed capacities (case discussion for red / blue edges)
  - at any vertex v, incoming flow still equals outgoing flow (if v is not on the path, nothing changes, else case discussion $\times$ 4)
- **the value increases**
  - the path must have a single edge containing s, and this is edge is blue
- **integer flow:** x is an integer

Ford and Fulkerson's algorithm

Max Flow algorithm

- initialize the flow with all values at 0
- while possible, do the improvement step

Claim

The algorithm computes a maximal flow

Proof: will take some work

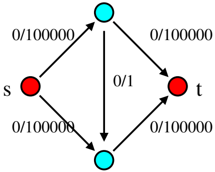
Claim

Runtime is $\boldsymbol{O(mM)}$, where M is the maximal value of the flow.

Proof: each improvement step costs $O(m)$ and increases the value by **at least 1** (integers!), so we can do at most M improvement steps.

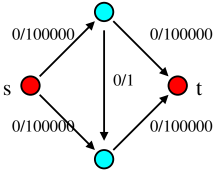
An example of a slow calculation

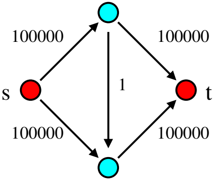
flow **residual graph** **augmenting path**



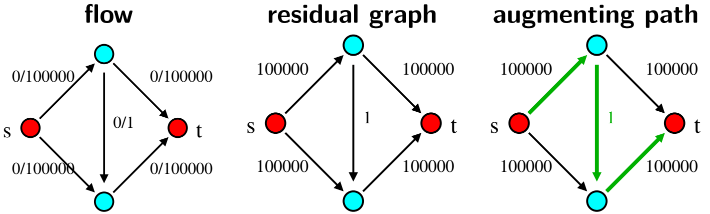
An example of a slow calculation

flow **residual graph** **augmenting path**

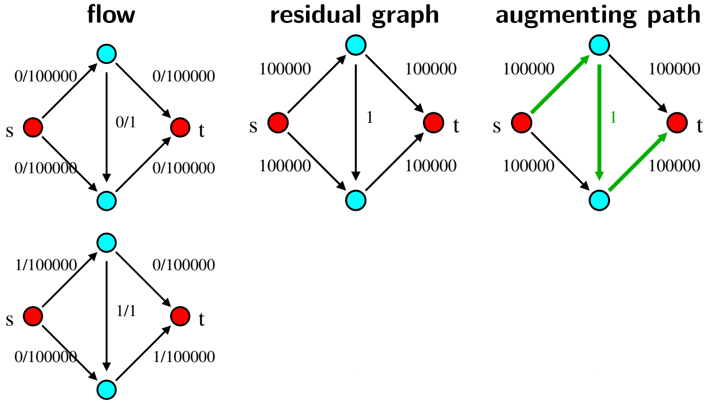




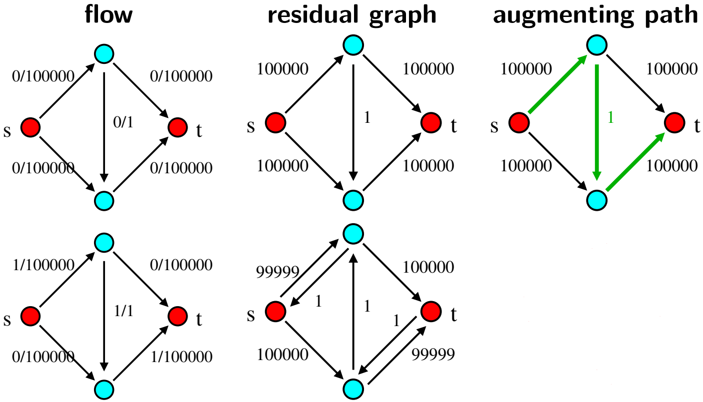
An example of a slow calculation



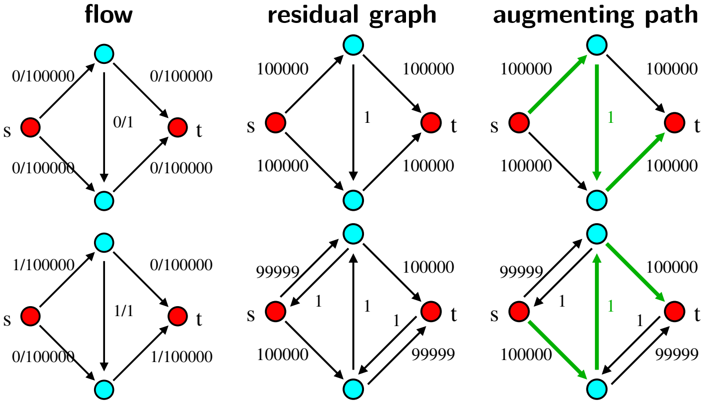
An example of a slow calculation



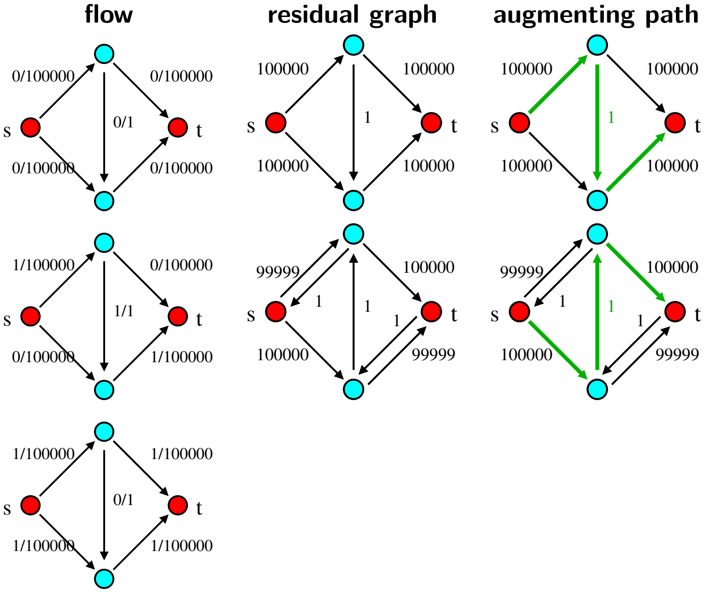
An example of a slow calculation



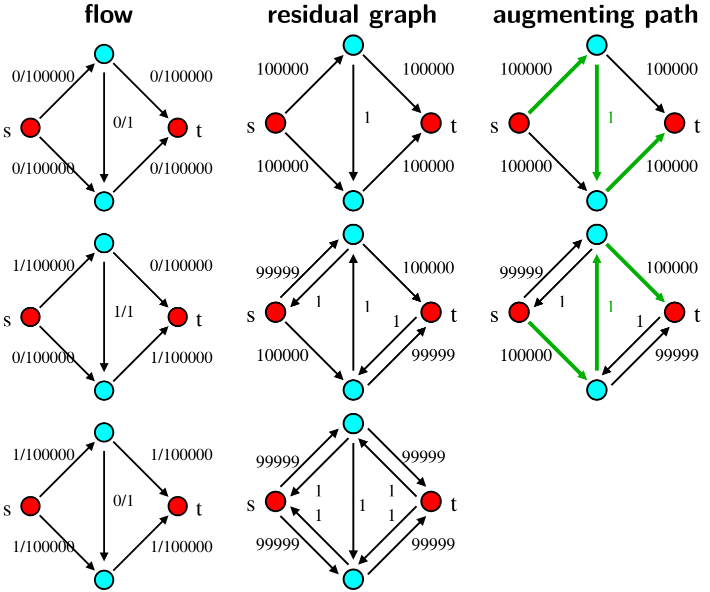
An example of a slow calculation



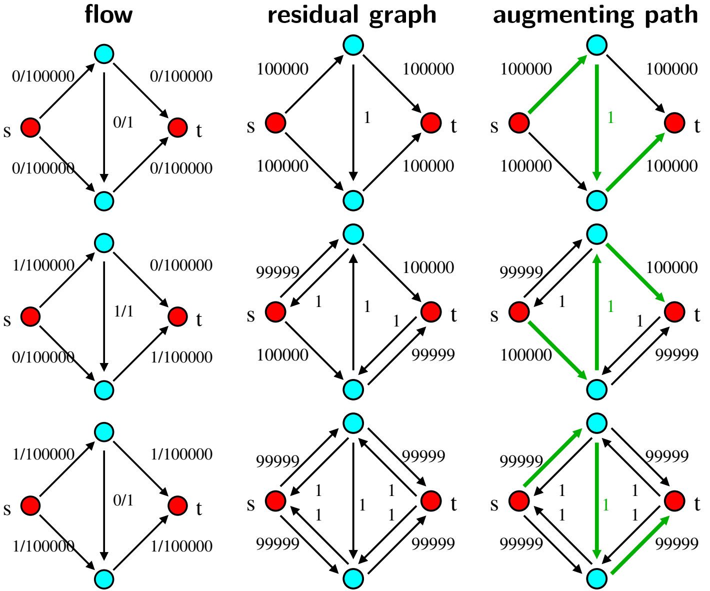
An example of a slow calculation



An example of a slow calculation

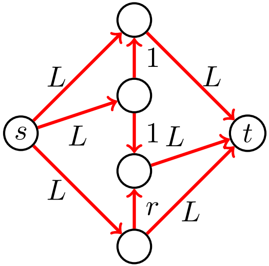


An example of a slow calculation



Integer capacities needed for termination

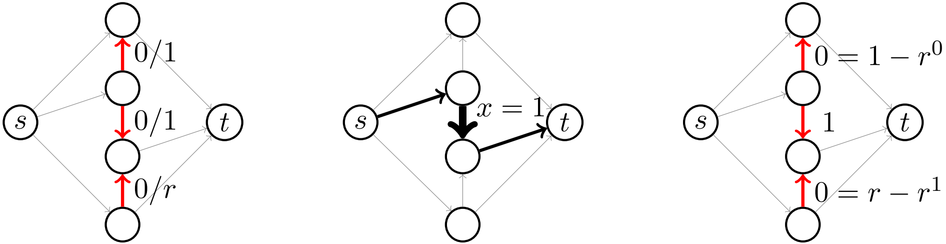
let $r = (\sqrt{5} - 1)/2 \simeq 0.618$, L a large integer and consider this graph:



Observations

- easy to find a flow of value $2L + 1$
- this is the best we can do (max flow = min cut, next lecture)
- but Ford-Fulkerson may loop forever

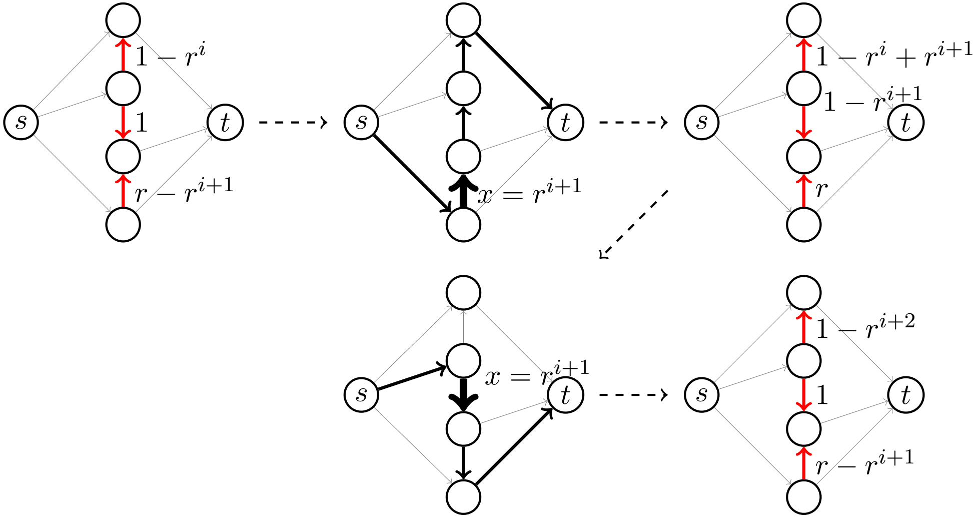
Initialization



Remarks:

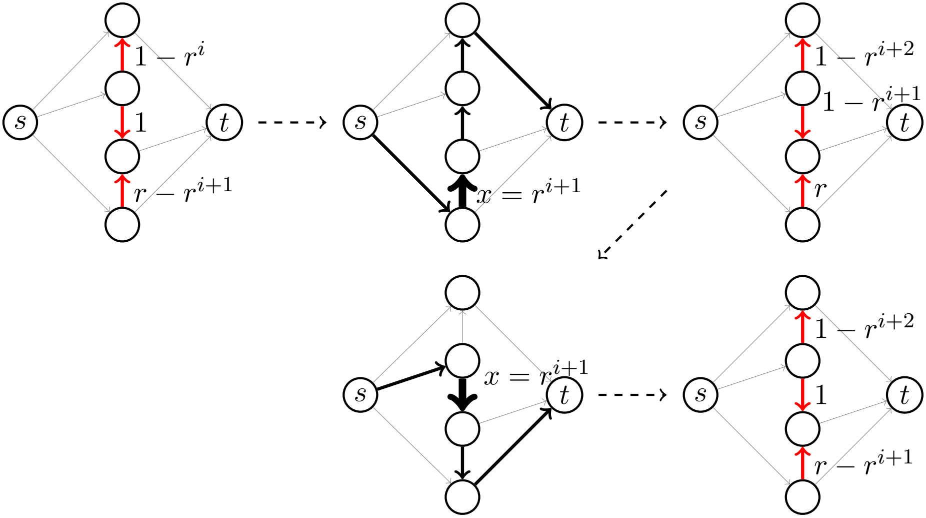
- flow on edges from s and edges to t not shown:
  - large capacity,
  - never a bottleneck
- value of the flow so far: **1**

Two augmentation steps



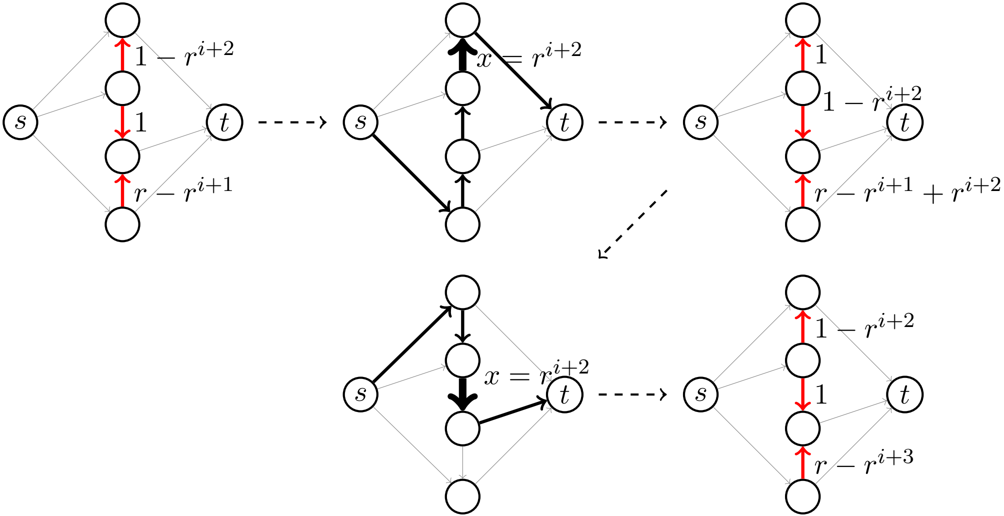
Flow increases by $\boldsymbol{2r^{i+1}}$

Two augmentation steps



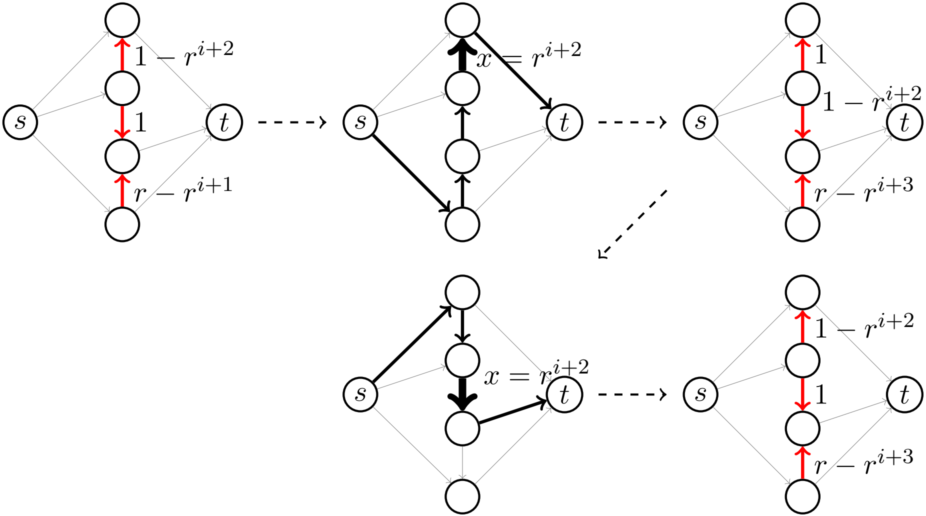
Flow increases by $\boldsymbol{2r^{i+1}}$

Another two augmentation steps



Flow increases by $\boldsymbol{2r^{i+2}}$, and we are back to the previous step with $i \leftarrow i + 2$

Another two augmentation steps



Flow increases by $\boldsymbol{2r^{i+2}}$, and we are back to the previous step with $i \leftarrow i + 2$

Conclusion

Regarding Ford-Fulkerson's algorithm

- may loop forever, with value approaching

$$\mathbf{1 + 2 \sum_{i \geq 1} r^i = \sqrt{5} + 2}$$

- optimal flow is $2L + 1$ (L large)

Computing with irrational numbers?

- computing with powers of r **feasible**:

$$r^i = \frac{a_i}{b_i} + \frac{c_i}{d_i}\sqrt{5}, \quad a_i, b_i, c_i, d_i \text{ integers}$$

  can be added, multiplied, compared

- but assuming that a_i, b_i, c_i, d_i fit in a word is **irrealistic**, a_i, b_i are $\Theta(\text{golden ratio}^i)$