CS 163 & CS 265: Graph Algorithms

Week 6: Cliques and coloring

Lecture 6c: Coloring, register allocation, Strahler number, and interval graphs

David Eppstein
University of California, Irvine

Winter Quarter, 2024

This work is licensed under a Creative Commons Attribution 4.0 International License

Register allocation and Strahler number

The register allocation problem

A compiler converts...

- Input: Code you wrote, in a high-level language, using however many local variables is convenient to express your ideas

$\Longrightarrow$

- Intermediate code: Every subexpression like the "$(x + y)$" in the expression $(x + y) \times z$ becomes a separate local variable

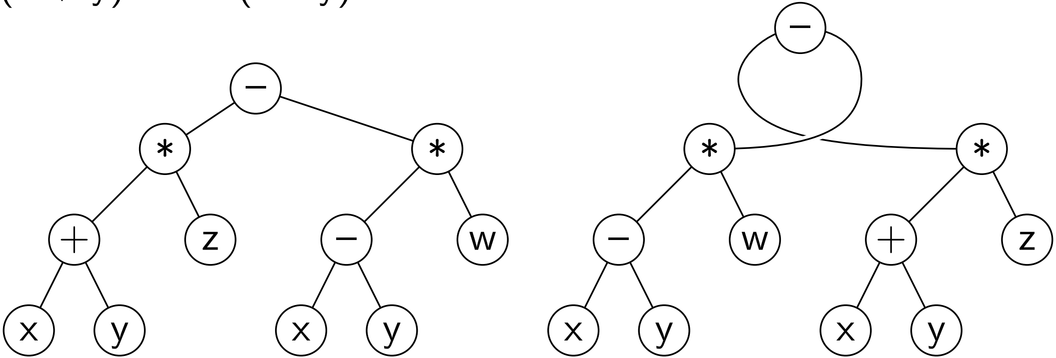
$\Longrightarrow$

- Machine code: Only a very small number of local variables can be stored in general-purpose machine registers (e.g. eax, ebc, ecx, edx); everything else is "spilled" into memory and more expensive to access

Special case: optimally ordering subexpressions

We can choose order to evaluate subexpressions of an expression (even when some operations are not commutative):

$(x + y) * z - (x - y) * w$:



Which ordering can be compiled into code with fewest registers?

Evaluation of orderings

Suppose we have already calculated

- Subexpression A can be evaluated using a registers
- Subexpression B can be evaluated using b registers

and we want to combine them into a single expression with one more operation

- If we calculate A first, and save it in a register while we calculate B, we need $\max(a, b+1)$ registers
- If we calculate B first, we need $\max(b, a+1)$ registers
- It never helps to mix the two calculations

We should try to perform trickier computations first
so that the simpler parts are the parts that get the $+1$ penalty

Bottom-up calculation of optimal ordering

For each node x of the expression tree, let $R(x)$ be the optimal number of registers needed to calculate the subexpression rooted at x

- If x is a leaf then $R(x) = 1$
- If x has two children y and z with $R(y) \neq R(z)$, then we want to evaluate the subtree with the larger R first, giving $R(x) = \max(R(y), R(z))$.
- If x has two children y and z with $R(y) = R(z) = r$, then it doesn't matter which order we evaluate them; either order gives $R(x) = r + 1$.

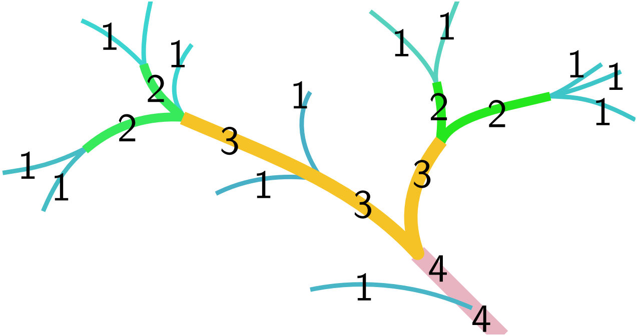
[Ershov 1958; Flajolet et al. 1979]

Strahler number

We can repeat the same calculation abstractly on any rooted tree, not just expression trees and not just binary trees:

- At leaf nodes, $R(x) = 1$
- At nodes where one child y has larger R than all other children, $R(x) = R(y)$
- At nodes where two or more children have the max value r, $R(x) = r + 1$

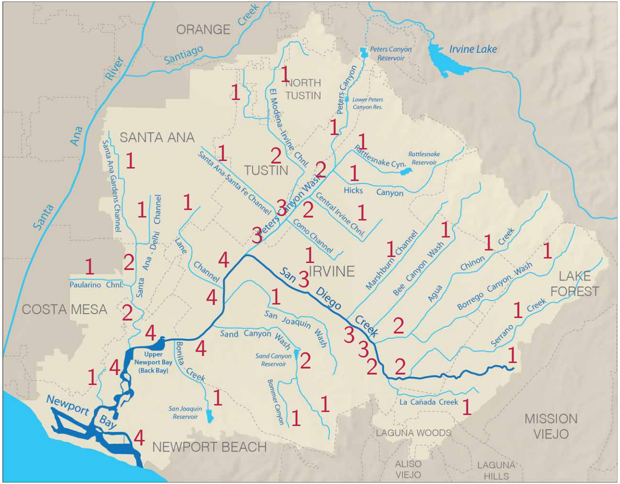
Called Strahler number or Horton–Strahler number



Original application of Strahler numbers

Evaluating the complexity of river networks in geography

[Horton 1945; Strahler 1952, 1957]

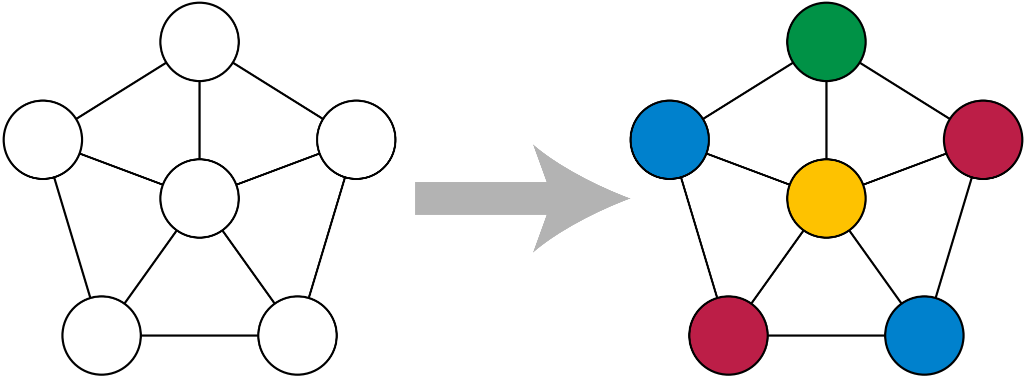


Graph coloring

Graph coloring

Input: Undirected graph

Output: Assignment of colors to vertices so that each edge has different colors at its endpoints, using as few colors as possible

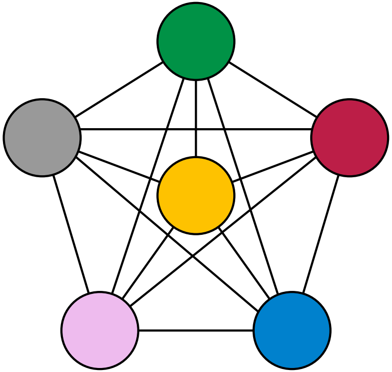


Special case: graphs needing only two colors are called “bipartite”

The four color theorem

If you can draw it without crossings in the plane,
you can color it using at most four colors
[Appel and Haken 1977]

But many other graphs require many more than four colors



Complexity of graph coloring

It's NP-hard and hard to approximate

Testing whether 2 colors is possible is easy (bipartiteness)
but testing whether 3 colors is possible is NP-complete,
even for graphs that can be drawn without crossings

Trivial approximation algorithm with approximation ratio $n/3$:
if it's not bipartite, just give every vertex a different color

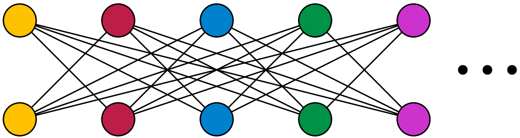
For every $\varepsilon > 0$ it is NP-hard to approximate better than $n^{1-\varepsilon}$
[Zuckerman 2007]

Greedy coloring heuristic

Since we can't guarantee an approximation ratio, use a method that always finds a coloring but might be far from optimal

Greedy coloring:

- Order the vertices (somehow)
- Number the available colors 1, 2, 3, 4, . . .
- For each vertex in the given ordering, give it the lowest-numbered available color (the smallest number that is not already used for a neighbor)



A bad example: remove a matching from a complete bipartite graph and order the vertices by matched pairs

Greedy coloring time analysis

Assuming the ordering has already been found (somehow)

Main remaining step: Find first unused color at each vertex

```
def first_unused(v):
    used = { color(w) for w in neighbors of v }
    for c in 1, 2, ...
        if c not in used: return c
```

Time: $O(\text{degree})$ per vertex, $O(m)$ total

Same "minimum excluded value" computation also comes up frequently in combinatorial game theory

Coloring by degeneracy

Suppose G has degeneracy d

Use greedy coloring algorithm with the reverse of a degeneracy order (so each vertex has $\leq d$ earlier neighbors, rather than later neighbors)

$\Rightarrow$ uses at most $d + 1$ colors

This is why degeneracy is also called coloring number

Register allocation and graph coloring

Register allocation as a graph coloring problem

Vertices: The local variables of the intermediate code

Edges: Two local variables that both need to be stored at the same time (their lifetimes, obtained from use-definition analysis, overlap)

Colors: Available registers

Try to color as much of the graph as possible
using # colors = # registers,
and then spill the remaining variables

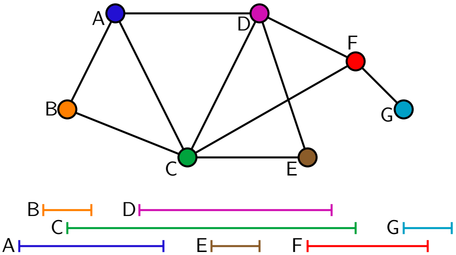
Register assignment for straight-line code

Suppose that we have code that is just expressions and assignments

no if-then-else, no loops, no subroutines

Expression trees already been ordered, so we just have a sequence of instructions $\text{var}_i = \text{var}_j$ op var_k, one assignment/variable

Lifetime of each variable is an interval from assignment to last use

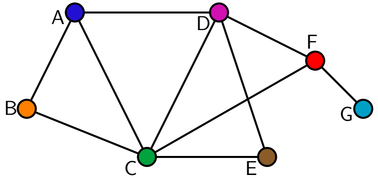


Graph of conflicting vars (overlapping intervals) is an interval graph

Greedy coloring for interval graphs

Order vertices left-to-right by left endpoint

Give each vertex first available color (standard greedy coloring)



A: first free color is 1

B: color is 2 (A has 1)

C: color is 3 (A,B have 1,2)

D: color is 2 (A has 1)

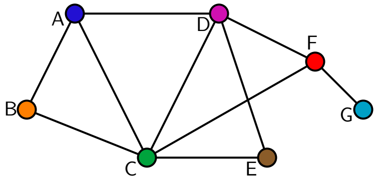
E: color is 1

F: color is 1

G: color is 2 (F has 1)

Total: 3 colors used

Greedy is optimal for interval graphs



If the algorithm uses color k for interval X, then:

- X already has neighbors colored $1, 2, \ldots k - 1$
- Because these neighbors are already colored, they have earlier left endpoints, and all overlap X at the left endpoint of X
- They also overlap each other at the same point
- We have a clique, a set of vertices all adjacent to each other
- Within the clique, we can only use one color per vertex
- So the whole graph needs at least k colors

The morals of the story

Graph coloring is hard, and hard to approximate

Applications include register allocation in compilers

Two easy special cases for register allocation:
optimally ordering expression trees (Strahler number), and
straight-line code (greedy coloring of interval graphs)

References I

Kenneth Appel and Wolfgang Haken. Solution of the four color map problem. *Scientific American*, 237(4):108–121, October 1977. doi:10.1038/scientificamerican1077-108.

A. P. Ershov. On programming of arithmetic operations. *Communications of the ACM*, 1(8):3–6, 1958. doi:10.1145/368892.368907.

P. Flajolet, J. C. Raoult, and J. Vuillemin. The number of registers required for evaluating arithmetic expressions. *Theoretical Computer Science*, 9(1):99–125, 1979. doi:10.1016/0304-3975(79)90009-4.

Robert E. Horton. Erosional development of streams and their drainage basins: hydro-physical approach to quantitative morphology. *Geological Society of America Bulletin*, 56(3):275–370, 1945. doi:10.1130/0016-7606(1945)56[275:EDOSAT]2.0.CO;2.

References II

Shannon1. Map of the Newport Bay watershed. CC-BY-SA image, January 11 2016. URL https://commons.wikimedia.org/wiki/File:NEWPORT_WATERSHED_MAP.png.

Arthur Newell Strahler. Hypsometric (area-altitude) analysis of erosional topology. *Geological Society of America Bulletin*, 63(11):1117–1142, 1952. doi:10.1130/0016-7606(1952)63[1117:HAAOET]2.0.CO;2.

Arthur Newell Strahler. Quantitative analysis of watershed geomorphology. *Transactions of the American Geophysical Union*, 38(6):913–920, 1957. doi:10.1029/tr038i006p00913.

D. Zuckerman. Linear degree extractors and the inapproximability of Max Clique and Chromatic Number. *Theory of Computing*, 3:103–128, 2007. doi:10.4086/toc.2007.v003a006.