Greedy Algorithms

Application: Optimal Caching

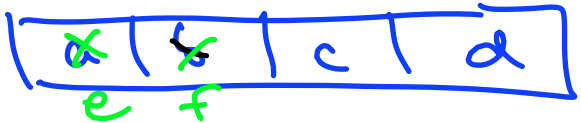
Algorithms: Design and Analysis, Part II

The Caching Problem

- small fast memory (the cache)
- big slow memory
- process sequence of "page requests"
- on a "fault" (that is, a cache miss), need to evict something from cache to make room – but what?

Example

Cache:



Request sequence: c d e f a b

=> 4 page faults

- 2 were inevitable (e & f)
- 2 consequences of poor eviction choices (should have evicted c & d instead of a & b)

The Optimal Caching Algorithm

Theorem: [Belady 1960s] the "furthest-in-future" algorithm is optimal (i.e., minimizes the number of cache misses).

Why useful?: ① serves as guideline for practical algorithms (e.g., least-recently-used (LRU) should do well provided data exhibits locality of reference).
② serves as idealized benchmark for caching algorithms

Proof: tricky exchange argument. **Open Question:** find a simple proof!

Tim Roughgarden