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: | ~~a~~ | ~~b~~ | c | d |

e f

Request sequence: c d e f a b

$\Rightarrow$ 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: [Bélády 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!