# CS186 Discussion Section 7
# Aggregation and Query Optimization
# 10/22/2013

## Review: Sorting & Hashing

a) Consider a simple multi-pass external merge sort algorithm that in the first pass, produces disk-based runs of “B” pages where B is the number of pages of memory available to the algorithm. If B = 50 pages, how many passes (including pass 1) of the algorithm will be required to sort a relation of size 5000 pages and how many runs will be produced by each of the passes?

**Total Passes = 1 + ceil($\log_{50-1}$(5000 / 50)) = 1 + $\log_{49}$100 = 3**
**Pass 1: 5000 / 50 = 100 runs**
**Pass 2: Merge B-1=49 runs at a time: ceil(100 / 49) = 3 runs**
**Pass 3: Merge 49 runs at a time: ceil(3 / 49) = 1 run**

b) Suppose we use the in memory sort optimization that generates runs of (on average) length 2B during pass 1. If B = 50, how many passes are required (including pass 1) and how many runs will be produced by each of the passes?

**Initial runs are of length 100.**
**Total Passes = 1 + ceil($\log_{49}$(5000 / 100)) = 3**
**Pass 1: 5000 / 100 = 50 runs**
**Pass 2: ceil(50 / 49) = 2 runs (49-way merging)**
**Pass 3: ceil(2 / 49) = 1 run (49-way merging)**

c) In what scenarios might we prefer to use hashing instead of sorting?

**GROUP BYs / Aggregations with small # of distinct values (hashing scales with # distinct values, not #items)**

**To take advantage of extra memory using hybrid hashing and perform fewer passes over the data.**

## Aggregations/Query Optimization

Consider the following schema:

Car(license, owner_ssn, year, company, model)
Accident(license, accident_date, damage_amount, zipcode)
Owner(ssn, license, name, gender, street, city, zipcode)

NTuples(Car) = 1000 ;          NPages(Car) = 100
NTuples(Accident) = 500 ;    NPages(Accident) = 20
NTuples(Owner) = 800 ;        NPages(Owner) = 50
NDistinct(Car.company) = 50;

For the following questions, assume that we have B=20 pages of available in-memory buffer space.

a) For the query: "SELECT gender, COUNT(*) FROM Owner GROUP BY gender;" What is the IO cost for aggregation by sorting?

**sort Owner: 4*50 = 200 IOs**
**scan and aggregate = 50 IOs**
**total = 250 IOs**
**w/ optimization, we can get this down to 150 IOs (combine the last pass of sorting with the scan/aggregate step).**

b) For the query in part a), what is the IO cost for aggregation by hybrid hashing?

**Since gender has very few group values, the count for each gender will fit in memory. You only need to scan through the Owner table once, to compute the query. 50 IOs**

c) For the query: "SELECT * FROM Accident A, Car C WHERE A.license = C.license AND A.damage_amount > X;" For what types of values of X would selection push-down significantly improve the cost of the query (Car is the inner table of the join)?''

**Selection push-down will help with very large values of X, since that would be more selective, and thus result in fewer resulting tuples for the rest of the plan.**

d) For the query: "SELECT O.name FROM Car C, Owner O WHERE C.license = O.license AND C.company = 'Volvo';" What is the expected cardinality of the Car relation after the initial selections are applied?

**You can only push down the Car.company = 'Volvo' selection predicate.**
**NDistinct(Car.company) = 50, so we can estimate Selectivity(Car.company) = 1/50.**
**Cardinality(Car.company = 'Volvo') = Selectivity(Car.company) * NTuples(Car) = 1000 / 50 = 20**