Sample solutions

Question 1.

a) 100/20 = 5

b) 100/20*10 = 50

c) probability of a tuple having b=2 in W is 1/60
since there are 5 tuples for a = 10,
=> for a=10 and b=2, we should have 1/60*5 = 0.083

d) probability of a tuple having b>2 in W is 58/60.
=> answer = 5*58/60 = 4.8

e) for c = 20, X is expected to have 200/100 = 2 tuples

we note that every tuple of Y will join with some tuple of X, but not the reverse. in fact, we expect 50% of X to have no matching Y records on c.

if there is a match, it will produce 6 tuples per match; otherwise 0.

so, can either be 12 tuples or 0.

f) (W Join X) by b = (100*200)/max(V(W,b),V(X,b)) = 200*100/60

Further join with Y on c, (200*100/60)*300/100 = 1000

Question 2.

Query result size = 72*2*100 + 30*40*9 + 10*16*7 + 4*250*9 + 4*2*20 + 95*(4*2*9)
= 14400 + 10800 + 1120 + 9000 + 160 + 6840 = 42320 tuples

Result has 2 attributes; so 15 tuples fit in a page. Therefore, in terms of pages, we need $\lceil 42320/15 \rceil$ = 2822 pages.

Question 3.

a. The cheapest initial join is R2 JOIN R3, with a result size of 100*100/10 = 1000. (actually, this is tied for cheapest, so the greedy algorithm might make this choice.) Next, we could join R2 JOIN R3 with either R1 or R4; the result size is the same, at 1000*1000/100 = 10,000. Thus, the greedy algorithm produces a plan with a cost of 11,000.

b. However, the globally best strategy (not necessarily restricted to left deep trees) is to start by joining R1 JOIN R2, and R3 JOIN R4, each of which have a cost of 1000*100/max(100,10) = 1000. This plan has a cost of 2000.

Again, here, we see that the best strategy in a restricted space may be far from the global optimal.

Question 4.

Note: You can have different answers if you store the intermediate results. However, the optimal solution (in terms of minimum I/O cost) is as follows.

(a) Solution: The total cost of this query plan is 119 I/Os computed as follows:

- (1) The cost of scanning Applicants is 100 I/Os. The output of the selection operator is 100/20 = 5 pages or 2000/20 = 100 tuples.
- (2) The cost of scanning Schools is 10 I/Os. The selectivity of the predicate on rank is $(10-1)/100 = 0.09$. The output is thus $0.09*10 \approx 1$ page or $0.09*100 = 9$ tuples.
- (3) Sort-merge join. Given that the input to this operator is only six pages, we can do an in-memory sort-merge join. The cardinality of the output will be 9 tuples. Consider that this is a key-foreign key join and each applicant can match with at most one school. However, keep in mind that the predicates on city and rank were independent, hence we have 100 tuples all with unique sid. So, only 9 tuples will end-up with a matching school.
- (4) Index-nested loops join. The index-nested loop join must perform one look-up for each input tuple in the outer relation. We can assume that each student only declares a handful of majors, so all the matches fit in one page. The cost of this operator is thus 9 I/Os.
- (5) and (6) are done on-the-fly, so there are no I/Os associated with these operators.

(b) System R employs a number of heuristics: avoid cross products, search only left deep space.

Solution:

Many solutions were possible including: