**201433707 이형욱**

**12.1**

We will refer to the tuples (kangaroo, 17) through (baboon, 12) using tuple numbers t1 through t12. We refer to the jth run used by the ith pass, as rij. The initial sorted runs have three blocks each. They are:

r11 = {t3,t1,t2} r12 = {t6,t5,t4} r13 = {t9,t7,t8} r14 = {t12,t11,t10}

Each pass merges three runs. Therefore the runs after the end of the ﬁrst pass are:

r21 = {t3,t1,t6,t9,t5,t2,t7,t4,t8} r22 = {t12,t11,t10}

At the end of the second pass, the tuples are completely sorted into one run:

r31 = {t12,t3,t11,t10,t1,t6,t9,t5,t2,t7,t4,t8}

**12.3**

r1 needs 800 blocks, and r2 needs 1500 blocks. Let us assume M pages of memory. If M > 800, the join can easily be done in 1500 + 800 disk accesses, using even plain nested-loop join. So we consider only the case where M ≤ 800 pages.

a. Nested-loop join: Using r1 as the outer relation we need 20000 ∗ 1500 + 800 = 30,000,800 disk accesses, if r2 is the outer relation we need 45000 ∗ 800 + 1500 = 36,001,500 disk accesses.

b. Block nested-loop join: If r1 is the outer relation, we need ⌈ 800 M−1⌉ ∗ 1500 + 800 disk accesses, if r2 is the outer relation we need ⌈ 1500 M−1⌉ ∗ 800 + 1500 disk accesses.

c. Merge-join: Assumingthatr1 andr2 are not initially sorted on the join key, the total sorting cost in clusive of the output is B s = 1500(2⌈logM−1(1500/M)⌉+

**Exercises 3**

2) + 800(2⌈logM−1(800/M)⌉ + 2) disk accesses. Assuming all tuples with the same value for the join attributes ﬁt in memory, the total cost is Bs + 1500 + 800 disk accesses. d. Hash-join: We assume no overﬂow occurs. Since r1 is smaller, we use it as the build relation and r2 as the probe relation. If M > 800/M ,i.e.no need for recursive partitioning, then the cost is 3(1500+800) = 6900disk accesses, else the cost is 2(1500+800)⌈logM−1(800)−1⌉+1500+800 disk accesses.

**12.6**

a. Use the index to locate the ﬁrst tuple whose branch city ﬁeld has value “Brooklyn”. From this tuple, follow the pointer chains till the end, retrieving all the tuples. b. For this query, the index serves no purpose. We can scan the ﬁle sequentially and select all tuples whose branch city ﬁeld is anything other than “Brooklyn”. c. This query is equivalent to the query

s(branch city≥′Brooklyn′ ∧ assets<5000)(branch)

Using the branch-city index, we can retrieve all tuples with branch-city value greater than or equal to “Brooklyn” by following the pointer chains from the ﬁrst “Brooklyn” tuple. We also apply the additional criteria of assets < 5000 on every tuple.

**12.10**

**12.18**