On the Cardinality Predicting Problems of Set Operations

Kohei Noshita[†] and William F. McColl

The cardinality predicting problems on various set-theoretical operations such as intersection, projection and join, are proposed and their computational complexity is investigated in terms of the number of comparisons on two types of the computing model. Those results are all optimal within constant factors. An approximation algorithm for computing the cardinality of join is also presented.

1. Introduction

In this paper, certain computational complexity problems concerning primitive operations on finite sets are defined and investigated. Those problems may be called "cardinality predicting problems," or in short "predicting problems," because of the way in which they are supposed to be applied, as explained below. The predicting problem, for instance, for intersection is to decide, for given two sets and a threshold integer, whether the cardinality of intersection of those two sets is no more than the threshold. The main purpose of this paper is to derive the optimal complexity for solving the predicting problems for various set operations.

Those set operations considered here appear, for example, in the relational database system ([1], [9]), in which the predicting problem may be of practical importance. In this system, we often want to know in advance the size of the result of primitive operations, because it may crucially affect the computing time for moving data as well as the amount of the output to the external devices. Hence it is important to design efficient algorithms for predicting whether the size of the result is acceptable to the predetermined threshold. In this situation, those algorithms are required to run much faster than those for actually obtaining the final result itself. Although in the relational database system the basic operations are defined as operating on tuples, in our context those operations considered here may be easily reformulated in terms of sets and multisets.

In this paper, we consider decision trees [3] as the model of algorithms to solve the predicting problems, and derive the asymptotic complexity in terms of the number of comparisons between two elements on two different types of decision trees. They are the linear-order model and the equal-unequal model. Both models have been discussed in the literature in order to study the complexity of various types of problems on sets as well as multisets. See [2], [4], [7] and [8].

The results obtained in this paper are summarised as follows. In case of such operations as intersection, difference, union and join, each corresponding predicting problem has the same complexity as the problem for computing the entire result of the operation. This gives us the negative result in the sense that there is no asymptotically faster algorithm for prediction. On the other hand, it is shown that, in case of the projection operation, the fast prediction is possible. The optimal complexity for the projection operation is determined within a constant factor. Finally, an approximation algorithm for computing the cardinality of join is presented and its complexity is analysed.

2. Basic Definitions and Predicting Problems

We shall deal with various operations on finite sets, which are assumed to be subsets of some fixed universal set Z of sufficiently large size. Two different types of Z will be considered with respect to its ordering relation; namely, Z is assumed either to be linearly ordered or to have no explicit order.

For such operations as projection and join, multisets as a generalised notion of sets will be considered [3], where a multiset is defined to be a set in which elements with the identical value may appear more than once. The ordinary symbols to represent operations on set will be used on multisets as well. When we particularly need to distinguish multisets from sets, the symbols $\langle$ and $\rangle$ will be used to denote a multiset, in place of the standard symbols $\{$ and $\}$ for a set.

Definition

For any multiset S, let $p(S)$ denote the set of distinct elements in S. For example, $p(\langle 1, 2, 1, 3\rangle)=\{1, 2, 3\}$.

Definition

For two multisets S and T,

join $(S, T)=\langle(a, b)|a$ in S, b in T and $a=b\rangle$.

For example, if $S=\langle 1, 2, 1, 3\rangle$ and $T=\langle 1, 1, 2\rangle$, then join $(S, T)=\langle(1, 1), (1, 1), (2, 2), (1, 1), (1, 1)\rangle$.

Now we are ready to list up our problems to be investigated in the following sections.

(1) intersection ($\cap$): For two sets S, T and an integer

Department of Computer Science University of Warwick Coventry CV4 7AL United Kingdom.

*This work was supported by SRC grant No. GR/A/9418.8 of United Kingdom.

†Present address Department of Computer Science Denkitusin University Chofu Tokyo 182 Japan.

h, decide whether $\# (S \cap T) \leqslant h$ or not.

In a similar way, the following two problems may be defined.

(2) difference $(-)$: $\# (S-T) \leqslant h$.

(3) union $(\cup)$: $\# (S \cup T) \leqslant h$.

(4) projection (proj): For a multiset S and an integer h, decide whether $\# p(S) \leqslant h$ or not.

(5) join: For two multisets S, T and an integer h, decide whether $\#$ join$(S, T) \leqslant h$ or not.

As mentioned in Sec. 1, in the relational database system, the operations listed above, among others, are defined as operating on sets of tuples. In our context, those sets of tuples may be regarded as sets or multisets, by ignoring all the columns of tuples that are not directly manipulated by the operation in question. For example, assume that S is a set of tuples

$$\{(1, a), (1, b), (2, c)\}$$

and that the first column is to be projected. Then, by ignoring all the second columns, the multiset $\langle 1, 1, 2\rangle$ may be defined to be an object for projection. Thus we have $p(\langle 1, 1, 2\rangle) = \{1, 2\}$. For other operations, the similar reformulation in terms of sets and multisets may now be straightforward.

Throughout this paper, all the algorithms will be described by means of decision tree models. For general definitions, see Knuth [3]. For our problems, either the ternary branching or the binary branching will be used depending on the assumption on Z. In the first case on the "linear-order" set Z, the comparison $a: b$ is used to determine whether $a > b$ or $a = b$ or $a < b$, for any a and b in Z. In the second case on Z without any order, the comparison $a: b$ gives only "equal-unequal" information that either $a = b$ or $a \neq b$, for any a and b in Z. As usual, the complexity of an algorithm is defined to be the height of the corresponding decision tree, i.e., the maximum number of comparisons. In this paper, we consider only the worst case complexity.

Definition

$C_\cap(m, n, h)$ will denote the number of comparisons required to solve the predicting problem for intersection for two sets S and T with $m = \# S$ and $n = \# T$ and an integer h. $C_\cap(n, h)$ will be used to denote an abbreviated form of $C_\cap(n, n, h)$.

Similarly, the notations C_-, $C_\cup$, C_{proj} and C_{join} are defined. Although two models are dealt with separately, no new notations to distinguish them will be introduced, because the model being used can be clearly understood from the context throughout the paper.

For notational convenience, the symbol $<\cdot$ will denote the asymptotic inequality when we ignore all terms of lower order than the leading term. For example, if $f(n) = n^2$ and $g(n) = 10n^2 - 100n$, then $f(n) <\cdot g(n)$. Now the meaning of the symbols $\leqslant\cdot$ and $=\cdot$ may be clear. The base of log is assumed to be 2.

3. Complexity on the Linear-Order Model

In this section, we shall derive the complexity for each predicting problem on the linear-order model. For our proof, we need the following lemma, whose proof is found in [7]. See also [4] and [8] for the generalised version of the lemma.

Lemma 3.1

The problem to decide, for any two sets S and T with $n = \# S = \# T$, whether $S \cap T = \phi$ has the complexity of $\theta(n \log n)$. The problem for $S = T$ has also the same complexity.

Note that more generally the complexity for $S \cap T = \phi$ with $m = \# S$ and $n = \# T$ $(m \leqslant n)$ is $(n+m) \log m$ with lower terms. Similarly, the complexity for $S \subseteq T$ may be shown to be $(n+m) \log m$ with lower terms.

Proposition 3.2

For any two sets S and T with $n = \# S = \# T$ and any integer h $(0 \leqslant h < n)$,

$$C_\cap(n, h) \text{ is } \theta(n \log n).$$

Proof

It is easy to see that the upper bound $O(n \log n)$ may be achieved by the obvious algorithm, which sorts S and T and merges them into a single sorted list.

The lower bound $\Omega(n \log n)$ may be proved by the "padding" argument. Let $X = S \cup \{a\}$ and $Y = T \cup \{b\}$, where $a, b \notin S \cup T$ and $a \neq b$. Apply the optimal algorithm B for $n+1$ and h to the input X, Y and h. Since $\#(X \cap Y) \leqslant h$ iff $\#(S \cap T) \leqslant h$, we can regard B as an algorithm A for n and h. Obviously, A is not faster than the optimal algorithm for n and h.

Hence we have

$$C_\cap(n, h) \leqslant C_\cap(n+1, h).$$

By a similar argument, we have

$$C_\cap(n, h) \leqslant C_\cap(n+1, h+1).$$

If $h \geqslant n/2$, the first inequality can be applied repeatedly, leading to:

$$C_\cap(n, h) \geqslant C_\cap(h+1, h).$$

Similarly, if $h < n/2$, the second inequality leads to:

$$C_\cap(n, h) \geqslant C_\cap(n-h, 0).$$

By Lemma 3.1, we have

$$C_\cap(n, h) = \Omega(h \log h) \quad \text{if } h \geqslant n/2 \text{, and}$$
$$C_\cap(n, h) = \Omega((n-h) \log (n-h)) \quad \text{if } h < n/2. \quad \square$$

Note that algorithm A in the proof may be viewed in a slightly different way. A faithfully minics B, except in the following situation. A ignores all the comparisons that involve any $(n+1)$-th element in X or Y, and regards two $(n+1)$-th elements as being distinct when those elements are compared in B.

A similar argument can be applied to obtain the complexity of C_- and $C_\cup$. We can prove the following propositions, by noting that, for any S and T with $n = \#S = \#T$,

$$\#(S-T)=0 \text{ iff } S=T,$$
$$\#(S-T)=n \text{ iff } S \cap T=\phi,$$
$$\#(S \cup T)=n \text{ iff } S=T, \text{ and}$$
$$\#(S \cup T)=2n \text{ iff } S \cap T=\phi.$$

Proposition 3.3

$$C_-(n, h)=\theta(n \log n),$$

where $0 \leqslant h < n$.

Proposition 3.4

$$C_\cup(n, h)=\theta(n \log n),$$

where $n \leqslant h < 2n$.

Proposition 3.5

$$C_{\text{proj}}(n, h)=\theta(n \log h),$$

where $1 < h < n$.

The proof of this proposition is found in the accompanying paper [6], in which the adversary (oracle) argument is applied.

Proposition 3.6

$$C_{\text{join}}(n, h)=\theta(n \log n),$$

where $0 \leqslant h < \cdot n^2$.

Proof

The upper bound may be achieved by a straightforward algorithm. See [2] and [4] for sorting algorithms for multisets.

Let S and T be any given two sets, as particular instances of multisets, with $n = \#S = \#T$. Clearly,

$$\#\text{join}\,(S, T)=0 \text{ iff } S \cap T=\phi, \text{ and}$$
$$\#\text{join}\,(S, T)=n \text{ iff } S=T.$$

By the padding argument used in the proof of proposition 3.2, we have

$$C_{\text{join}}(n, h)=\Omega(n \log n)$$

provided that $0 \leqslant h < n$.

Consider the case for $n \leqslant h < \cdot n^2$.

Let S and T be two sets with $n = \#S = \#T$. Append k elements with the identical value a to S, where a is not in S. Let X denote this new multiset; namely,

$$X = S \cup \langle a, a, \cdots, a\rangle.$$

Similarly, let Y denote the new multiset $T \cup \langle a, a, \cdots, a\rangle$, formed by appending k a's, where a is not in T. By applying the predicting algorithm for $n+k$ and $h+k^2$ to X and Y, we can obtain the desired result for S and T, because

$$\#\text{join}\,(X, Y) = \#\text{join}\,(S, T)+k^2.$$

Hence we have

$$C_{\text{join}}(n, h) \leqslant C_{\text{join}}(n+k, h+k^2).$$

Rewriting this inequality leads to

$$C_{\text{join}}(n, h) \geqslant C_{\text{join}}(n-k, h-k^2).$$

Here we choose the value of k to be

$$k = \lfloor \sqrt{h} \rfloor.$$

Since $n-k \geqslant h-k^2$, we can apply the result above, yielding the following lower bound.

$$\begin{aligned} C_{\text{join}}(n, h) &= \Omega((n-k) \log (n-k)) \\ &= \Omega((n-\sqrt{h}) \log (n-\sqrt{h})). \end{aligned} \qquad \square$$

The lower bound given in the proof is valid even for h very near n^2. However, the gap from the upper bound becomes large as h approaches n^2. Thus we need another consideration for $h = \cdot n^2$. In the extreme case, we have

$$C_{\text{join}}(n, h) = 2n-1$$

for $n^2 - n < h < n^2$.

Note that $\#\text{join}(S, T)$ with $n = \#S = \#T$ takes the values, in the decreasing order, as follows:

$$n^2, n^2-n, n^2-2n+2, n^2-2n+1, \cdots.$$

More generally, see the results in [5]. In this paper, we shall not examine those extreme cases any more. For a related result, see Sec. 5.

Note that, by generalising the argument above, we can derive the complexities of $C_\cap(m, n, h)$, $C_-(m, n, h)$, $C_\cup(m, n, h)$ and $C_{\text{join}}(m, n, h)$. They are all $\theta(n \log m)$ for $m \leqslant n$.

4. Complexity on the Equal-Unequal Model

In this section, the complexity of the predicting problems is derived on the equal-unequal model. The following lemma is used in our proof.

Lemma 4.1 [7]

The problem to decide, for any two sets S and T with $n = \#S = \#T$, whether $S \cap T = \phi$ has the complexity of $\theta(n^2)$. The problem for $S = T$ has also the same complexity.

Note that the constant factor of this complexity is exactly one. The proof of this lemma is found in [7], in which the results are presented in the more general way. See also [8].

We list up the results in this section.

All the propositions may be proved by means of the similar proof techniques used in the previous section, except the proof of proposition 4.5 which is included in the accompanying paper [6]. Thus, we omit the proof of all the results.

Proposition 4.2

$$C_\cap(n, h) = \theta(n^2),$$

where $0 \leqslant h < n$.

Proposition 4.3

$$C_-(n, h) = \theta(n^2),$$

where $0 \leqslant h < n$.

Proposition 4.4

$$C_\cup(n, h) = \theta(n^2),$$

where $n \leqslant h < 2n$.

Proposition 4.5 [6]

$$C_{\text{proj}}(n, h) = \theta(nh),$$

where $0 < h < n$.

Proposition 4.6

$$C_{\text{join}}(n, h) = \theta(n^2),$$

where $0 \leqslant h < \cdot n^2$.

With regard to C_{proj}, the exact number of comparisons can be determined for certain specific values of h; namely,

$$C_{\text{proj}}(n, 1) = n-1$$
$$C_{\text{proj}}(n, 2) = 2n-3,$$
$$C_{\text{proj}}(n, n-2) = (n+1)(n-2)/2, \text{ and}$$
$$C_{\text{proj}}(n, n-1) = n(n-1)/2.$$

The lower bounds for $h=2$ and $n-2$ are not obvious. Each proof needs the appropriate construction of the oracle, which involves rather complicated case analyses. Since the proof is too lengthy to be included in this paper, it is left to the interested reader.

Note that the results for $C_\cap$, C_-, $C_\cup$ and C_{join} may be generalised to the case with three parameters m, n and h. They are all $\theta(mn)$.

5. Computing the Cardinality of Join

In this section we present an approximation algorithm to compute the cardinality of join on the linear-order model.

Let X and Y be two multisets of n elements. In the description of the algorithm, p will denote some constant depending on $n(1 \leqslant p < n)$. For notational brevity, let $S < T$ denote that the relation $s < t$ holds for any s in S and t in T. Similarly, $S \leqslant T$ is defined.

begin

Step 1:

Divide X into $(p+1)$ elements $\langle a_0, a_1, \cdots, a_p \rangle$ and p multisets $S_1, S_2, \cdots, S_p$, such that

$$\{a_0\} \geqslant S_1 \geqslant \{a_1\} \geqslant \cdots \geqslant S_p \geqslant \{a_p\}$$

and

$$\#S_i = (n-p-1)/p \text{ for } 1 \leqslant i \leqslant p.$$

This partial sorting may be achieved by means of the linear-time median-selection algorithm [3], applied recursively to divide a multiset into two smaller multisets of almost equal size. Thus, the cost of this step is bounded by $O(n \log p)$.

Step 2:

By scanning through X, for each $a_i (0 \leqslant i \leqslant p)$, compute its 'multiplicity'; namely, find all the elements with the value equal to a_i. This yields the following refinement of X:

$$A_0 > T_1 > A_1 > T_2 > \cdots > T_q > A_q,$$

where $q \leqslant p$ and A_j is a multiset of elements with the value equal to some a_i $(0 \leqslant i \leqslant p, 0 \leqslant j \leqslant q)$.

Note that $\#T_i \leqslant (n-p-1)/p$. It is easy to see that this step may be achieved in $O(n)$ comparisons.

Step 3:

Throughout this step, the index i $(0 \leqslant i \leqslant q)$ runs in such an order as $\lfloor q/2 \rfloor$, $\lfloor p/4 \rfloor$, $\lfloor 3q/4 \rfloor$, $\lfloor q/8 \rfloor, \cdots$. Namely, the next value of i is chosen to be the middle value of the range in order that it is divided into two smaller ranges of almost equal size.

For each i, compare a_i with elements in Y, so that Y is divided into $(q+2)$ sets $R_j (0 \leqslant j \leqslant q+1)$ and $B_i = \langle b | b = a_i \rangle$ $(0 \leqslant i \leqslant q)$, where

$$R_0 > B_0 > R_1 > B_1 > \cdots > B_q > R_{q+1}.$$

Note that R_j as well as B_i may be empty. Taking the order of index i into account, the cost of this step can be proved to be at most $O(n \log q)$.

end

Define

$$E = u_0 \times v_0 + u_1 \times v_1 + \cdots + u_q \times v_q,$$

where $u_i = \#A_i$ and $v_i = \#B_i$ for $0 \leqslant i \leqslant q$. Note that $u_i > 0$ and $v_i \geqslant 0$. The output of the algorithm is the value of E.

Proposition 5.1

$E \leqslant \#\text{join}(X, Y) \leqslant E + n^2/p$, and E can be computed in $O(n \log p)$ comparisons.

Proof

The number of comparisons has been counted in the description of the algorithm. The first inequality is obvious. The second inequality may be derived by noting the following relation:

$$\#T_1 \times \#R_1 + \cdots + \#T_q \times \#R_q$$
$$\leqslant (n/p) \times (\#R_1 + \#R_2 + \cdots + \#R_q) \leqslant n^2/p. \quad \square$$

As direct corollaries of this proposition, we look at some examples of the relative error of E. Let $p = (\log n)^k$ for some constant $k (\geqslant 1)$. If $\#\text{join}(X, Y)$ is

(1) $n^2/(\log n)^k$,
(2) $n^2/\log \log n$, or
(3) cn^2 for $0 < c < 1$,

then the relative error of E is, respectively,

(1) 1,
(2) $\log \log n/(\log n)^k$, or
(3) $1/c(\log n)^k$.

The complexity of all three cases is $O(n \log \log n)$. Note that the complexity for computing $\#\text{join}\ (X, Y)$ is $\theta(n \log n)$.

We can also show that, for any fixed ε $(0<\varepsilon<1/3)$, if there exists an approximation algorithm A with output E such that $|\#\text{join}(X, Y)-E| \leqslant \varepsilon(\#\text{join}(X, Y))$ for any X and Y, then the complexity of A is $\Omega(n \log n)$.

6. Concluding Remarks

In this paper, we proposed the predicting problems on set operations and derived their asymptotic complexities in terms of the number of comparisons on the linear-order model as well as on the equal-unequal model. An approximation algorithm to compute the cardinality of join was also presented.

As final remarks, we suggest several related problems for future study. With regard to the primitive operations appearing in the relational database system, the quotient operation is to be considered, where quotient $(\div)$ is defined as follows: For two sets S and T such that $S \subseteq Z \times Z$ and $T \subseteq Z$ $(T \neq \phi)$,

$$S \div T = \{x \mid (x, y) \text{ in } S \text{ for any } y \text{ in } T\}.$$

Although our complexity results in Sec. 3 and Sec. 4 are all optimal within constant factors in terms of the asymptotic functional growth, it may be natural to pose the problem to determine the exact number of comparisons for each operation. However, the necessity of the complicated case analyses in proving the lower bounds for $C_{\text{proj}}(n, 2)$ and for $C_{\text{proj}}(n, n-2)$ in Sec. 4 suggests the difficulty of this problem.

Finally, the predicting problem may be reformulated on different types of computing model in order to make it more directly applicable to real situations. For example, a system based on the hashing technique seems worthwhile to investigate as a more practical model. In fact, the hashing technique is reported to have been used for prediction in some practical systems. Another possible model for our study may be a computing model with limited random access storage.

References
1. Codd, E. F. A Relational Model of Data for Large Shared Data Banks, *Comm. ACM*, **13**, 6 (1970), 377-387.
2. Dobkin, D. and Munro, J. I. Determining the Mode, *Theoretical Computer Science*, **12**, 3 (1980), 255-253.
3. Knuth, D. E. The Art of Computer Programming, **3**, *Sorting and Searching*, Addison Wesley, Reading, MA (1973).
4. Munro, J. I. and Spira, P. M. Sorting and Searching in Multisets, SIAM J. Computing, **5**, 1 (1976), 1-8.
5. McColl, W. F. and Riha, W. O. *(m, n)*-Expressibility and the Join Operation, Technical Report No. 129, Department of Computer Science, University of Leeds (1979), 18 pp.
6. Noshita, K. Predicting the Number of Distinct Elements in a Multiset, to appear in SIAM J. Computing, **11**, 4 (1982).
7. Reingold, E. M. On the Optimality of Some Set Algorithms, J. *ACM*, **19**, 4 (1972), 649-659.
8. Stockmeyer, L. J. and Wong, C. K. On the Number of Comparisons to Find the Intersection of Two Relations, SIAM J. *Computing*, **8**, 3 (1979), 388-404.
9. Ullman, J. D. Principles of Database Systems, Pitman, London (1980).

(Receivd February 5, 1982: revised June 14, 1982)