$\phi(t)$ is the state occupied at time 't', and ϕ_m and ϕ_n are any two states with transition probabilities

$$\text{Prob}\,[\phi(t+1) = \phi_n \,|\, \phi(t) = \phi_m] = q_{m,n},$$

then, $P^*(\phi_m)$ and $P^*(\phi_n)$, the equilibrium probabilities of being in ϕ_m and ϕ_n respectively, obey:

$$P^*(\phi_m)/P^*(\phi_n) = q_{n,m}/q_{m,n} \quad \text{for all } m, n. \tag{5}$$

Now the transition from $(R_{i_1}, R_{i_2}, \ldots, R_{i_j}, R_{i_{j+1}}, \ldots, R_{i_N})$ to $(R_{i_1}, R_{i_2}, \ldots, R_{i_{j+1}}, R_{i_j}, \ldots, R_{i_N})$ occurs whenever $R_{i_{j+1}}$ is accessed from the left or R_{i_j} is accessed from the right. Similarly, the transition from $(R_{i_1}, R_{i_2}, \ldots, R_{i_{j+1}}, R_{i_j}, \ldots, R_{i_N})$ to $(R_{i_1}, R_{i_2}, \ldots, R_{i_j}, R_{i_{j+1}}, \ldots, R_{i_N})$ occurs whenever $R_{i_{j+1}}$ is accessed from the right or R_{i_j} is accessed from the left. Observing that these events are mutually exclusive and substituting them in (5), we have:

$$\frac{P^*(R_{i_1}, R_{i_2}, \ldots, R_{i_j}, R_{i_{j+1}}, \ldots, R_{i_N})}{P^*(R_{i_1}, R_{i_2}, \ldots, R_{i_{j+1}}, R_{i_j}, \ldots, R_{i_N})} = \frac{s_{i_j,L} + s_{i_{j+1},R}}{s_{i_{j+1},L} + s_{i_j,R}} \quad \text{for all } j. \tag{6}$$

Hence the theorem.

A similar result exists for the Transposition rule for the SLL.[12] Theorem II is a generalization of the latter result. Indeed, in the special case when the probability of access from the right is zero (that is $s_{i,R} = 0$ for all i), the Swap heuristic behaves identically to the Transposition heuristic and (6) becomes identical to the expression in Ref. 12:

$$\frac{P^*(R_{i_1} R_{i_2} \cdots R_{i_j} R_{i_{j+1}} \cdots R_{i_N})}{P^*(R_{i_1} R_{i_2} \cdots R_{i_{j+1}} R_{i_j} \cdots R_{i_N})} = \frac{s_{i_j,L}}{s_{i_{j+1},L}}$$

Rivest[12] also proved that the Transposition heuristic is always more efficient than the Move-To-Front heuristic. We can now state an analogous claim concerning the Swap heuristic.

Conjecture I

The Swap heuristic is always more efficient than the Move-To-End heuristic.

Experimental results clearly support the conjecture that this result is true for all distributions.[15] However, the proof of the claim is certainly non-trivial. It will involve obtaining expressions for the ratio of the asymptotic probabilities of the chain being in two states, say, ϕ_m and ϕ_n in which the list representation of ϕ_m can be derived by interchanging exactly two of the elements in ϕ_n and vice versa. Examples of two such states are the orderings

$$\phi_m = (R_{i_1}, R_{i_2}, \ldots, R_x, R_{k_1}, R_{k_2}, \ldots, R_{k_z}, R_y, \ldots, R_{i_N}),$$

and,

$$\phi_n = (R_1, R_{i_2}, \ldots, R_y, R_{k_1}, R_{k_2}, \ldots, R_{k_z}, R_x, \ldots, R_{i_N}).$$

In the above, the representation of the states ϕ_m and ϕ_n, are interchangeable by merely interchanging the position of the elements R_x and R_y. Computing such expressions for the transposition heuristic is straightforward because of the cancellations of the asymptotic probabilities of being in the intermediate states. But generalizing the latter for the case of the Swap heuristic is non-trivial because such cancellations do not occur. Furthermore, it is not certain whether such a generalization may even exist. If the latter is the case and such a generalization does not exist, the result will have to be proved using mathematical tools which have not been used in the literature to analyze adaptive data structures. These will involve not merely computing the asymptotic probabilities of the associated states but also deriving expressions for the asymptotic cost evaluated in terms of the asymptotic state occupancy probabilities.

To conclude this section, we also conjecture that if two SLL heuristics τ and ρ satisfy the property that τ is more efficient than ρ, then the corresponding pair of DLL heuristics, τ' and ρ', obtained using the directed mapping, are such that τ' is more efficient than ρ'. This conjecture is a generalization of both Theorem I and the above conjecture for the set of directed mappings. The result seems intuitive since a more efficient SLL scheme tends to push the frequently accessed elements to the end, and thus plausibly do a better job to polarize the elements in the DLL.

III. Conclusion

In this paper, we have studied the problem of adaptively reorganizing a Doubly-Linked List (DLL) and have presented an interesting relationship between arbitrary pairs of heuristics for this problem and those used to reorganize a Singly-Linked List (SLL). This is achieved by introducing the concept of undirected and directed mappings between SLLs and DLLs. In particular, we have introduced a new heuristic called the Swap heuristic, and derived the asymptotic probabilities for the chain converging to the various possible orderings. Two conjectures have been proposed which suggest possible directions for future research in the area of adaptive DLLs.

Acknowledgements

The authors are grateful to various readers for their comments and in particular to Mr Valiveti for helpful discussions.

D. T. H. NG AND B. J. OOMMEN
School of Computer Science,
Carleton University,
Ottawa, Ontario K1S 5B6, Canada

References

1. E. J. Anderson, P. Nash and R. R. Weber, A counter-example to a conjecture on optimal list ordering. *J. Appl. Prob.* **19** (3) 730–732 (1985).
2. J. R. Bitner. Heuristics that dynamically organize data structures, *SIAM J. Comput.* **8** (1), 82–110 (1979).
3. P. J. Burville and J. F. C. Kingman, On a model for storage and search, *J. Appl. Probability*, **10**, 697–701 (1973)
4. J. H. Hester and D. S. Hirschberg, Self-organizing linear search, *Comp. Surveys*, **17** (3), 295–311 (1985).
5. W. J. Hendricks, The stationary distribution of an interesting Markov chain, *J. Appl. Probl.* **9** (1) 231–233 (1972).
6. Y. C. Kan and S. M. Ross, Optimal list order under partial memory constraints, *J. Appl. Prob.* **17** (4) 1004–1015 (1980).
7. D. E. Knuth, *The Art of Computer Programming, Vol. 3: Sorting and Searching*, pp. 398–399. Addison-Wesley, Reading, Mass. (1973).
8. D. Matthews, D. Rotem and E. BretHolz, Self-organizing doubly-linked lists, *J. Comp. Maths.* A, **8** 99–106 (1980).
9. J. McCabe, On serial files with relocatable records, *Oper. Res.* 609–618 (1965).
10. B. J. Oommen and E. R. Hansen, List organizing strategies using stochastic move-to-front and stochastic move-to-rear operations, *SIAM J. of Comput.*, **16** (4) 705–716 (1987).
11. B. J. Oommen, E. R. Hansen and J. I. Munro, Deterministic optimal and expedient move-to-rear list organizing strategies. *Theoretical Computer Science*, **74**, pp. 183–197, (1990).
12. R. Rivest, On self-organizing sequential search heuristics, *C-ACM* **19** (2) 63–67 (1976).
13. S. M. Ross, *Introduction to Probability Models.* Academic Press, New York (1980).
14. J. P. Tremblay and P. G. Sorenson, *An Introduction to Data Structures with Applications*, McGraw-Hill, New York, 1976.
15. D. T. H. Ng and B. J. Oommen, Generalizing singly-linked list reorganizing heuristics for doubly-linked lists, *Proceedings of the 1989 Conference on the Mathematical Foundations of Computer Science, Rytro, Poland*, 380–389 (1989).

Indexing for multi-attribute retrieval

1. Introduction

In his book,[4] Professor Wiederhold discusses the concept of multi-attribute indexing: this involves maintaining an index on the concatenation of several attributes, to provide a faster response to a query specifying values for several different attributes. Since in a given query, an attribute may or may not be specified, it is necessary to maintain some index to serve every combination. (Here, an index is said to *serve* a subset of attributes, if it could be used to answer a query involving precisely that subset of attributes.)

An interesting feature of multi-attribute indexing is that one index can be used to answer queries for several different choices of subsets of specified attributes. For example, the index on attributes (1,2,3,4) concatenated in that order would serve the subsets {1}, {1,2}, {1,2,3}, and {1,2,3,4}. In fact, a multi-attribute index on k attributes could serve k different subsets: one of cardinality i for every i with $(1 \leqslant i \leqslant k)$. The actual subsets served are determined by the order of concatenation of the attributes. Wiederhold noted that by a clever choice of order, the number of indexes required for 4 attributes could be reduced from 15 to 6.

Since each index can serve precisely one subset of cardinality i, $(1 \leqslant i \leqslant k)$, it follows that $\binom{4}{2}$ is in fact the minimum number of indexes required to serve all combinations of 4 attributes. For 6 attributes, Wiederhold used trial and error to work out a scheme to serve all combinations with 20 indexes. His success at that led him to conjecture that the lower bound of $\binom{N}{\lceil \frac{N}{2} \rceil}$ could be achieved in the general case. He was not aware of any way to generate such a set of indexes and he posed it as an open question with a request for a solution.

An affirmative solution to this conjecture was presented informally, by the first author, at the Database Seminar of the Stanford Department of Computer Science. Professor Wiederhold and others present noted that, since the indexes used are not necessarily in lexicographic order, it would be useful to have a means of providing the proper order of concatenation for a subset specified in lexicographic order.

Our solution was incorporated into Professor Wiederhold's book *File Organization for Database Design* Ref. 5, p. 168. Related algorithms have been discussed in Refs 1–3.

The purpose of this note is to formulate and solve the same problem in a more formal setting. Our hope is to give a better insight into the combinatorial nature of this problem.

Section 2 proposes an algorithm to generate the set of indexes sought in Wiederhold's question. Section 3 gives an algorithm which will return the specific index generated by the algorithm of section 2 which serves a subset provided as input.

2. The main result

It was noted in the introduction that the order of the concatenation of attributes in a multi-attribute index determines which subsets of attributes that index serves. More precisely, the ordered r-tuple $X = (x_1, \ldots, x_r)$ of distinct integers $1 \leqslant x_i \leqslant N$ serves the family $F = \{\{x_1\}, \{x_1, x_2\}, \ldots, \{x_1, x_2, \ldots, x_r\}\}$ of subsets of $\{1, \ldots, N\}$. To simplify notation, in the algorithm Build_Index_Set attributes are represented by characters from an alphabet and indexes by strings of that alphabet. For the purpose of specifying the details of our algorithms, we shall find it convenient to rely on the following functions operating on strings:

LEN(A): Returns the length of string A; (an empty string has length 0).

CONCAT(A, B): Returns the string produced by appending *B* to the end of *A*.

LEFT (A, n): Returns the string consisting of the *n* leftmost characters of *A*.

```
Procedure Build_Index_Set(N);
{Input: an integer N;
Output: a minimum set R[1..$\binom{N}{\lceil N/2 \rceil}$] of indexes
that serve every subset of {1,...,N}.}
0.  begin
1.      R[1] ← 'Λ'; {here, Λ stands
            for the empty string}
2.      rpt ← 2;
3.      for i ← 1 to N do begin
4.          spt ← rpt − 1;
5.          for j ← 1 to spt do begin
6.              if LEN(R[j]) ≥ $\lceil \frac{i}{2} \rceil$ then begin
7.                  R[rpt] ← LEFT
                    (CONCAT('i', R[j]),
                    LEN(R[j]));
8.                  rpt < −rpt + 1;
9.              end;
10.             R[j] < −CONCAT(R[j], 'i')
11.         end
12.     end;
13.     return(R)
14. end;
```

We are now in a position to state our main result.

Theorem 1. *R is a minimal set of indexes which serves all subsets of* $\{1, \ldots, N\}$.

Our proof of Theorem 1 relies on several intermediate results which we present as lemmas.

To simplify our exposition, when referring to the procedure *Build_Index_Set* (and when no confusion is possible) the term *iteration* always refers to an iteration of the outer loop; furthermore, we shall let $S(p, q)$ stand for the set of all the strings of length q in R at the end of the pth iteration of *Build_Index_Set*. In particular, $S(N, t)$ denotes the set of all the strings of length t in R when *Build_Index_Set* terminates. We let $|A|$ stand for the cardinality of set A.

Lemma 1. *At the end of the kth iteration of procedure Build_Index_Set, R contains precisely* $\binom{k}{l} - \binom{k}{l+1}$ *strings of length l, for* $\lceil \frac{k}{2} \rceil \leqslant l \leqslant k$.

Proof of Lemma 1. By induction on k; the statement is trivially true at the end of the first iteration; assume the statement true at the end of the $(k-1)$th iteration of procedure *Build_Index_Set*.

Let 1 be an arbitrary, but fixed, integer in the range $\lceil \frac{k}{2} \rceil$ to k. To find $S(k, 1)$, we note that in iteration k of the outer loop for every t such that $\lceil \frac{k}{2} \rceil \leqslant t \leqslant k$.

- Line 10 replaces every string in $S(k-1, t-1)$ with a string in $S(k, t)$,
- Line 7 writes one new string in $S(k, t)$ for every string in $S(k-1, t)$ [since $t \geqslant \lceil \frac{k}{2} \rceil$, the condition of the if statement in 6 is met], and
- Every non-empty string in R was written in Line 7 or Line 10 of the current iteration.

Thus, by the inductive hypothesis, $|S(k, 1)|$ is

$$\binom{k-1}{1} - \binom{k-1}{1+1} + \binom{k-1}{1-1} - \binom{k-1}{1}.$$

which can be written as

$$\left(\binom{k-1}{1} + \binom{k-1}{1-1}\right) - \left(\binom{k-1}{1+1} + \binom{k-1}{1}\right).$$

Now, applying twice a well known combinatorial identity, we get

$$|S(k, 1)| = \binom{k}{1} - \binom{k}{1+1},$$

as claimed. ∎

Lemma 2. *When Build_Index_Set terminates, there is a mapping f from the subsets of* $\{1, \ldots, N\}$ *to the strings in R such that* $f(Y)$ *serves Y and* $LEN(f(Y)) \geqslant max\{|Y|, N - |Y|\}$.

Proof. We will define recursively a sequence of functions f_k such that f_n is the desired mapping.

First, let $f_1(\{1\}) =$ '1'. Next, for an arbitrary subset S of $\{1, \ldots, k\}$ $(k \geqslant 2)$, set $S' \leftarrow f_{k-1}(S - \{k\})$. Now $f_k(S)$ is defined as follows:

(i) If k is not an element of S, then $f_k(S) =$ CONCAT(S', 'k').

(ii) Else if LEN(S') $= |S| - 1$, then $f_k(S) =$ CONCAT(S', 'k')

(iii) Else $f_k(S) =$ LEFT(CONCAT('k', S'), $|S|$).

To prove that the mapping f_N has the desired properties, we use induction on N. Clearly, the statement holds for $N = 1$. Assume it holds for $N - 1$. We not that if $f_N(S)$ is assigned in (iii), then LEN(S') $\geqslant |S|$ and, since by the induction hypothesis LEN(S') $\geqslant$ max $\{|S| - 1, N - (|S| - 1)\}$ it follows that LEN(S') $\geqslant \lceil \frac{N}{2} \rceil$. Thus the condition of step 6 of procedure *Build_Index_Set* is met and the string defined in (iii) is written in step 7 of *Build_Index_Set*. Since LEN(S') $> |S - \{N\}|$, LEFT(S', $|S| - 1$) still serves $S - \{N\}$ and thus LEFT(CONCAT('N', S'), $|S|$) serves S. We also have LEN($f_N(S)$) $=$ LEN(S'). Since LEN(S') $\geqslant |S|$ and $N - |S| = N - 1 - |S - \{N\}|$, the length condition is still met.

If $f_N(S)$ is assigned in (i) or (ii), then the string assigned is the index that replaced S' in step 10 of procedure *Build_Index_Set*, and so, LEN($f_N(S)$) $=$:LEN(S') $+ 1$, as claimed. ∎

Proof of Theorem. Let T be an arbitrary set of indexes which serves every subset of $\{1, \ldots, N\}$. We shall find it convenient to rely on two straightforward observations whose justification is immediate.

Observation 1. *A subset Y of* $\{1, \ldots, N\}$ *can only be served by an index of T of length at least* $|Y|$.

Observation 2. *No two distinct subsets Y, Z of* $\{1, \ldots, N\}$ *of the same cardinality can be served by the same index in T.*

By Lemma 2, R serves all subsets of $\{1, \ldots, N\}$. To prove the minimality of R, we only need associate with every index i_R in R, a corresponding index i_T in T, such that LEN(i_R) $\leqslant$ LEN(i_T), and such that distinct indexes in R are associated with distinct indexes of T.

To avoid using a fussy formalism, we shall describe the above mapping by the following procedure.

Step 1. Start with all indexes in T 'unmarked'; set $t \leftarrow 0$;

Step 2. Pick $\binom{N}{t} - \binom{N}{t-1}$ unmarked indexes in T of length at least $N - t$, and assign them to Q.

Step 3. Map every index i in $S(N, N-t)$ to a distinct index $j(i)$ in Q; j is 'marked' in T.

Step 4. Set $t \leftarrow t + 1$; if $t < \lceil \frac{N}{2} \rceil$ then goto Step 2 else exit.

Fact 1. *If T contains M marked indexes, there are at least* $\binom{N}{t} - M$ *unmarked indexes of length* $N - t$ *in T.*

Proof of Fact 1. By Observation 2, there are at least $\binom{N}{N-t} - M$ subsets of $\{1, \ldots, N\}$ of cardinality $N - t$ that cannot be served by the marked indexes in T. Now Observation 1 and 2 combined, guarantee that there are at least $\binom{N}{t} - M$ unmarked indexes in T of length at least $N - t$. ∎

Fact 2. *When Step 2 is about to be executed for the kth time, T contains* $\binom{N}{k-1}$ *marked indexes.*

Proof of Fact 2. The proof is by induction on k; the statement is obviously true for $k = 0$. Assume it holds true after k executions of Step 2. We only need prove that the statement holds true when Step 2 is about to be executed for the $(k+1)$th time.

Since there are, by the induction hypothesis $\binom{N}{k-1}$ marked indexes in T, Fact 1 guarantees that there must exist at least $\binom{N}{k} - \binom{N}{k-1}$ unmarked indexes in T of length at least

$N-k$. By Lemma 1, precisely $\binom{N}{k}-\binom{N}{k-1}$ of them get marked in Step 3, and the conclusion follows. ■

To complete the proof of Theorem 1 we only need observe that, by Fact 1 and Fact 2 combined, at the completion of the above procedure, each element of R will be mapped to an index of T of at least the desired length. ■

3. Generating an Individual Index

The following procedure takes the incidence vector of a subset of $\{1, ..., N\}$ as input and returns the index generated by procedure *Build_Index_Set* which serves that subset.

```
Procedure Index_Gen(Y);
{Input: An integer N and a subset Y of {1, ..., N} given by its incidence vector S
Output: The index generated by the procedure Build_Index_Set which serves Y}
0.  begin
1.    for i ← 1 to N do begin
2.      if S[i] = 0 then
3.        B ← Concat(B, 'i');
4.      else
5.        if B = Λ then
6.          A ← CONCAT(A, 'i');
7.        else begin
8.          A ← CONCAT('i', A);
9.          B ← LEFT(B, LEN(B) − 1);
10.       end
11.   end;
12.   return(CONCAT(A, B))
12. end;
```

Theorem 2. *For every subset Y of $\{1, ..., N\}$, Index_Gen returns the index generated by Procedure Build_Index_Set which serves Y and is of length at least $Max\{|Y|, N-|Y|\}$.*

Proof of Theorem 2. Let $S_k = (s_1, ..., s_k)$ and let Y_k be the subset of $\{1, ..., k\}$ whose incidence vector is S_k. We shall prove by induction on k that after k iterations of the loop of Procedure *Index_Gen*, $I_k = \text{CONCAT}(A, B)$ contains an index generated by k iterations of *Build_Index_Set* which is of length at least $\text{Max}\{|Y_k|, k-|Y_k|\}$ and that the string A contains precisely the elements of Y_k. The statement clearly holds for $k = 1$. Suppose it holds for $k-1$.

If $s_k = 0$, then only line 3 of *Index_Gen* executed in this iteration. By the induction hypothesis, $I_k = \text{CONCAT}(I_{k-1}, 'k')$, an index that would be written in Step 10 of procedure *Build_Index_Set*. Since $Y_k = Y_{k-1}$, the string A still contains precisely the elements of Y_k. Since $\text{LEN}(I_k) = \text{Len}(I_{k-1})+1$, the length requirement is met.

Now suppose $s_k = 1$. If the string B is empty, then only step 6 of *Index_Gen* will be executed. Now the string A contains precisely the elements of Y_k. As above, $I_k = \text{CONCAT}(I_{k-1}, 'k')$, thus the index is written by step 10 of *Build_Index_Set* and satisfies the length requirement. If the string B is non-empty, then steps 8 and 9 of *Index_Gen* are executed: Step 8 guarantees that the string A contains precisely the elements of Y_k; Step 9 guarantees that $I_k = \text{LEFT}(\text{CONCAT}('k', I_{k-1}), \text{LEN}(I_{k-1}))$. If $\text{LEN}(I_{k-1}) > \lceil\frac{k-1}{2}\rceil$, then I_k will be written in Step 7 of *Build_Index_Set* and will meet the length requirement. But $\text{LEN}(I_{k-1})$ must be greater than $\lceil\frac{k-1}{2}\rceil$, for otherwise $\text{Max}\{|Y_{k-1}|, k-|Y_{k-1}|\} = \lceil\frac{k-1}{2}\rceil$, thus $|Y_{k-1}| = \lceil\frac{k-1}{2}\rceil$ and B is empty. ■

B. JAMISON and S. OLARIU

Department of Computer Science, Old Dominion University, Norfolk, Virginia, 23529, U.S.A.

References

1. D. E. Knuth, *The Art of Computer Programming, Vol. 3: Sorting and Searching.* Addison-Wesley, Reading (1973).
2. V. Y. Lum, Multi-Attribute Retrieval with Combined Indexes. *CACM 13*, 660–665 (1970).
3. B. Shneiderman, *Reduced Combined Indexes for Efficient Multiple Attribute Retrieval. Information Systems 2* 149–154 (1977).
4. G. Wiederhold, *Database Design*, McGraw-Hill, New York, 189 (1983).
5. G. Wiederhold, *File Organization and Database Design*, McGraw-Hill, New York, 186 (1987).

Announcement

25–28 May 1993

The 8th Scandinavian Conference on Image Analysis, University of Tromsø, Norway

Announcement and first call for papers

Invitation to the 8th SCIA

The 8th Scandinavian Conference on Image Analysis will be arranged by the Norwegian Society for Image Processing and Pattern Recognition (NOBIM) and sponsored by the International Association for Pattern Recognition (IAPR). The conference will be held in Tromsø from 25 to 28 May 1993. Tromsø, located at latitude 69°40′ N, is Northern Norway's centre for administration and education.

Scientific programme

The scientific programme will include several invited speakers and parallel sessions with papers for oral and poster presentation. The conference language will be English. The contributed papers will cover original unpublished research results, either theoretical or applied.

Conference topics

- Image processing and analysis
- Pattern recognition
- Computer vision
- Parallel algorithms and architectures
- Neural nets
- Remote sensing
- Medical and biological applications
- Industrial applications

Submission of papers

Papers must be in English, and only full papers will be accepted. Authors are invited to submit three (3) copies of each full paper. The cover paper should contain: title of the paper; name(s) and affiliations of the author(s); brief abstract (2000 words); name and address for correspondence.

All pages should show the name of the first author and be consecutively numbered. In the final version all pictures must be rasterised. If the length of the paper exceeds 8 pages or contains colour illustrations an additional fee will be added to the registration fee.

The deadline for submission of papers is **15 October 1992.** All papers will be assessed by two reviewers. Authors will be notified as to acceptance by **20 December 1992.** For accepted papers, a final camera-ready copy will be required by **1 March 1993.** Instructions for writing the final paper will be sent to the authors.

Social programme

The social programme planned includes visits to Spitzbergen and North Cape. Participants and accompanying persons are invited to bring cross-country or mountain skiing equipment, as the skiing conditions in the mountains are normally superb in late May. Sea-fishing will also be arranged and we will have a good chance of enjoying all these activities in the midnight sun. We emphasise that in Tromsø, also called the gateway to the Arctic, the midnight sun lasts from approximately the 20th of May until the 20th of July. The scenery at the time of the conference will be snowy mountains and day-light the whole night through as the sun never goes to rest. The indoor activities will consist of a get-together party on the first night of the conference and the Conference Banquet, both of which are included in the conference fee (there is an extra fee for accompanying persons).

Further information

For all inquiries regarding the conference and/or for inclusion on the mailing list, please write to:

Kjell Arild Høgda, Forut Information Technology Ltd., P.B. 2806 Elverhøy, N-9001 Tromsø, Norway. Tel: +47 83 58 622. Fax: +47 83 82 420. E-mail: scia@conan.uit.no.

Requests of any kind concerning the conference or the social programme can be sent to this e-mail address. By simply mailing the text *send info* to **scia@conan·uit·no** you will automatically be included in our conference mailing list and will regularly be supplied with up-to-date information on the programmes.