1 References by Topic

1.1 Compression

Primary Texts

Khalid Sayood. *Introduction to Data Compression.* Morgan Kaufmann, 1996.

Other Sources

Timothy C. Bell, John G. Cleary, and Ian H. Witten. *Text compression.* Prentice Hall, 1990.

Gilbert Held and Thomas R. Marshall. *Data and Image Compression: Tools and Techniques.* Wiley, 1996 (4th ed.)

Mark Nelson. *The Data Compression Book.* M&T Books, 1995.

James A. Storer (Editor). *Image and Text Compression.* Kluwer, 1992.

Ian H. Witten, Alistair Moffat, and Timothy C. Bell. *Managing Gigabytes: Compressing and Indexing Documents and Images.* Van Nostrand Reinhold, 1994.

1.2 Cryptography

Primary Texts

Alfred J. Menezes, Paul C. van Oorschot and Scott A. Vanstone. *Handbook of Applied Cryptography.* CRC Press, 1996.

Bruce Schneier. *Applied Cryptography.* Wiley, 1996.

Douglas R. Stinson. *Cryptography: Theory and Practice.* CRC Press, 1995.

Other Sources

G. Brassard. *Modern Cryptology: a tutorial.* Spinger-Verlag, 1988.

N. Koblitz. *A course in number theory and cryptography.* Springer-Verlag, 1987.

C. Pfleeger. *Security in Computing.* Prentice-Hall, 1989.

A. Saloma. *Public-key cryptography.* Springer-Verlag, 1990.

Jennifer Seberry and Josed Pieprzyk. *Cryptography: An Introduction to Computer Security.* Prentice-Hall, 1989.

D. Welsh. *Codes and Cryptography.* Claredon Press, 1988.

1.3 Linear and Integer Programming

Texts

M. Bazaraa, J. Jarvis, and H. Sherali. *Linear Programming and Network Flows.* Wiley, 1990.

Dimitris Bertsimas and John N. Tsitsiklis. *Introduction to Linear Optimization.* Athena Scientific, 1997

J. P. Ignizio and T. M. Cavalier, *Linear Programming.* Prentice Hall, 1994

G. L. Nemhauser, A.H.G. Rinnooy Kan, and M. J. Todd (ed.). *Optimization.* Elsevier, 1989.

George L. Nemhauser and Laurence A. Wolsey. *Integer and combinatorial optimization.* Wiley, 1988.

H. M Salkin and K. Mathur. *Foundations of Integer Programming.* North-Holland, 1989.

Robert J. Vanderbei. *Linear Programming: Foundations and Extensions.* Academic Publishers, 1996

Stavros A. Zenios (ed.). *Financial optimization.* Cambridge University Press, 1993.

Useful Overview Papers

G. L. Nemhauser. *The age of optimization: solving large-scale real-world problems.* Operations Research, 42(1), Jan.-Feb. 1994, pp. 5-13.

E. L. Johnson and G. L. Nemhauser. *Recent developments and future directions in mathematical programming.* IBM Systems Journal, vol.31, no.1; 1992; pp. 79-93.

I. J. Lustig, R. E. Marsten and D. F. Shanno. *Interior point methods for linear programming: computational state of the art.* ORSA Journal on Computing, vol.6, no.1; Winter 1994; pp. 1-14.

1.4 Triangulation

Primary Texts

M. de Berg, M. van Kreveld M. Overmars, and O. Schwarzkopf. *Computational Geometry, Algorithms and Applications.* Springer-Verlag, 1997.

Other Texts

J. E. Goodman, and J. O'Rourke (Editors). *Handbook of Discrete and Computational Geometry.* CRC Press, 1997.

K. Mulmuley. *Computational Geometry: An Introduction Through Randomized Algorithms.* Prentice Hall, 1994.

J. Nievergelt and K. Hinrichs. *Algorithms and data structures: with applications to graphics and geometry.* Prentice Hall, 1993.

J. Pach and P.K. Agarwal. *Combinatorial Geometry.* John Wiley and Sons, 1995.

Preparata, Franco P. *Computational geometry: an introduction.* Springer-Verlag, 1988.

A. Okabe, B. Boots, and K. Sugihara. *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams.* John Wiley, 1992.

O'Rourke. *Computational Geometry in C.* Cambridge University Press, 1994.

1.5 N-body Simulation

Susanne Pfalzner and Paul Gibbon. *Many Body Tree Methods in Physics.* Cambridge University Press, 1997.

L. Greengard. *Fast algorithms for classical physics.* Science, vol. 265, no. 5174, 12 Aug. 1994, pp. 909-14.

J. E. Barnes and P. Hut. *A hierarchical O(N Log N) force calculation algorithm.* Nature, 324(4):446-449, December 1986.

1.6 VLSI Physical Design

Primary Texts

N. Sherwani. *Algorithms for VLSI Physical Design Automation, Second Edition.* Kluwer, 1995.

M. Sarrafzadeh and C. K. Wong. *An Introduction to VLSI Physical Design.* McGraw-Hill, 1996.

Other Texts

P. Banerjee. *Parallel Algorithms for VLSI Computer-Aided Design.* Prentice Hall, 1994.

A. B. Kahng and G. Robins. *On Optimal Interconnections for VLSI.* Kluwer Academic Publishers, 1995.

B. Korte, L Lovasz, H. J. Promel, and A. Schrijver (eds.). *Paths, flows, and VLSI-layout.* Springer Verlag, 1990.

T. Lengauer, *Combinatorial algorithms for integrated circuit layout.* Wiley, 1990.

M. Pecht and Y. T. Wong (editors). *Advanced Routing of Electronic Modules.* CRC Press, 1995.

M. Sarrafzadeh and D.T. Lee (editors). *Algorithmic Aspects of VLSI Layout (Lecture Notes Series on Computing, Vol 2).* World Scientific, 1994.

1.7 Pattern Matching in Biology

Dan Gusfield. *Algorithms on String, Trees, and Sequences.* Cambridge University Press, 1997.

Arthur M. Lesk. *Computational molecular biology: sources and methods for sequence analysis.* Oxford University Press, 1988.

Graham A Stephen. *String searching algorithms.* World Scientific, 1994.

M.S. Waterman. *Introduction to Computational Biology: Maps, Sequences, Genomes.* Chapman & Hall, 1995.

1.8 Indexing and Searching

Christos Faloutsos. *Searching Multimedia Data Bases by Content.* Kluwer Academic, 1996.

Frakes and Baeza-Yates (ed.). *Information Retrieval: Data Structures and Algorithms.* Prentice Hall, 1992

Frants V. J., Shapiro J., Voiskunskii V. G. *Automated Information Retrieval Theory and Methods.* Academic Press, Aug 1997.

Lesk M. *Practical Digital Libraries Books, Bytes & Bucks.* Morgan Kaufman Publishers, 1997.

Salton. *Automatic Text Processing: The Transformation, Analysis and Retrieval of Information by Computer.* Addison-Wesley, 1989.

Sparck Jones K. and Willett P. (editors). *Readings in Information Retrieval.* Morgan Kaufman Publishers, 1997.

Ian H. Witten, Alistair Moffat, and Timothy C. Bell. *Managing Gigabytes: Compressing and Indexing Documents and Images.* Van Nostrand Reinhold, 1994.

1.9 Clustering

Leonard Kaufman and Peter J. Rousseeuw. *Finding Groups in Data : An Introduction to Cluster Analysis.* Wiley, 1990.

Everitt, Brian. *Cluster analysis (3d ed.).* Wiley, 1993.

E. Backer. *Computer-Assisted Reasoning in Cluster Analysis.* Prentice Hall, 1995.

2 Assignments

Algorithms in the Real World
Assignment #1 (Compression) Due Date: 12 Sept 97

Collaboration policy: You are welcome to talk about this or other assignments with others and even discuss problems using a whiteboard or scratch paper. You are not allowed, however, to show or give any work that you did on your own pertaining to the assignment to someone else.

Problem 1: Assume an alphabet $S = \{a, b, c\}$ has the probabilities

$$p_a = .2 \quad p_b = .7 \quad p_c = .1$$

(a) What is the entropy $H(S)$?

(b) What is the average length of the Huffman code for S (remember to weigh the average based on the probabilities).

(c) What is the average length of the Huffman code if you group characters into pairs.

(d) For a long string what is the average per-character length of an arithmetic code for S (assume the asymptotic value and that the message abides by the probabilities).

(e) For a long string what is the average per-character length of an arithmetic code for S assuming your model are the probabilities given above, but you are coding a message which consists of just c's (i.e., your data does not fit your model).

(f) Given the following conditional probabilities what is the conditional entropy (this is the average per-character length that would be achieved by a good code that takes advantage of the probabilities).

$$\begin{array}{lll} p(a|a) = .25 & p(b|a) = .1 & p(c|a) = .8 \\ p(a|b) = .65 & p(b|b) = .8 & p(c|b) = .1 \\ p(a|c) = .1 & p(b|c) = .1 & p(c|c) = .1 \end{array}$$

The notation $p(x|y)$ is the probability of x given that the previous character is y.

Problem 2: Prove that the average length $l_a = \sum p_i l_i$ for a Huffman code for a set of messages S is bounded by the inequality $l_a \leq H(S) + 1$. To prove this you can assume the Huffman code is optimal. Note that given a maximum probability p_m a tigher upper bound of

$$l_a < \begin{cases} H(S) + p_m & p_m \geq .5 \\ H(S) + p_m + .086 & p_m < .5 \end{cases}$$

can be shown. Feel free to prove this stronger result.

Problem 3: Assume we are coding a 4 letter alphabet with LZW and we start with the dictionary containing the 4 letters $\{a, b, c, d\}$ in locations 0, 1, 2 and 3. Entries are added to the dictionary in sequential order. Decode the following message that was coded with LZW.

```
0 0 1 2 4 7
```

Problem 4: Using the LZSS (the variant of LZ77 described in the reading "Introduction to data compression" by Peter Gutmann) and assuming a window size W and a lookahead buffer size B, how many code words will it take to code a message of length L that contains all the same character. Assume each ¡offset,size¿ pair or character is a codeword. Also assume the window is initialized to some other character.

Problem 5: Lets say we had the following sequence of values

```
10 11 12 11 12 13 12 11 10 -10 8 -7 8 -8 7 -7
```

(a) Would we would get better compression by doing a cosine transform across the whole group of 16 or by blocking the 16 entries into two groups of 8 and transforming each separately? Explain briefly?

(b) If using a wavelet transform are we better off blocking the entries into two groups of 8?

Algorithms in the Real World
Assignment #2 (Cryptography) Due Date: 1 Oct 97

Problem 1:

Assume you design an encryption scheme with a single round DES (instead of the usual 16 rounds and ignoring the initial and final permutations). If you know a message cipher pair (m_1, c_1) for a key k for the values given below, what is m_2 assuming $c_2 = DES1_k(m_2)$. Assume the following hexidecimal values for m_1, c_1 and c_2.

$$\begin{aligned} m_1 &= \texttt{3F72D4E6 04CD825B} \\ c_1 &= \texttt{04CD825B 8E25E338} \\ c_2 &= \texttt{04CD825B 327C7D29} \end{aligned}$$

If you show the first 2 (leftmost) hexidecimal digits of m_2 that will convince me you know the rest.

Problem 2:

Assume we have a 3×3-bit s-box used in a block ciphers. For each of the following three substitutions explain what is wrong.

Input	out_1	out_2	out_3
0	2	0	4
1	5	4	3
2	3	1	7
3	7	5	2
4	0	2	5
5	4	6	0
6	3	3	6
7	1	7	1

Problem 3:

(a) Assuming p and q are primes, express $x^{-1} \bmod pq$ as a positive power of x.

(b) Given public key $e = 3$, $n = 391$ encode the value 45 using RSA.

(c) Assuming the same public key, decode the value 105 (you might want to write a short program to do it).

(d) In RSA for some n lets say $e = 3$ (a standard choice in some systems). If I accidentally give away d show how one can use this information to factor n.

Problem 4:

(a) Using ElGamal Encryption with the public key $p = 11, g = 7, y = 5$ encrypt the value 9 with random $k = 3$.

(b) Decrypt the pair (2,2).

(c) Given $p = 11$ why is 10 a bad value for g?

Problem 5:

(a) Generate the first 4 bits of the BBS random bit generator assuming that $n = 77, r_0 = 39$.

(b) For an arbitrary Blum integer $n = pq$ give an upper bound on the period of the generator (how many iterations it takes before it repeats). This need not be the strongest upper-bound but you will get more credit the stronger it is.

Algorithms in the Real World
Assignment #3 (Integer and Linear Programming) Due Date: 15 Oct 97

Please feel free to send me email if there are any ambigouous questions.

Problem 1: Strang Chapter 8 problem 8.1.9

Problem 2: Strang Chapter 8 problem 8.2.1.

Problem 3: Strang Chapter 8 problem 8.2.13.

Problem 4:

Consider the following problem

$$\begin{array}{ll} \text{maximize} & x_1 + x_2 \\ \text{subject to} & 2x_1 + x_2 \leq 4 \\ & x1, x2 \geq 0 \end{array}$$

(a) Starting with the initial solution $(x_1, x_2) = .1, .1$ solve this problem using the affine scaling interior-point method.

(b) Illustrate the progress of the algorithm on the the graph in $x_1 - x_2$ space.

Problem 5:

Let $S = \{x \in R^n : Ax \leq b\}$ and $T = \{x \in R^n : Bx \leq d\}$. Assuming that S and T are nonempty, describe a polynomial time algorithm (in n) for checking whether $S \subset T$.

Problem 6:

Formulate the following problem as a mixed integer programming problem.

$$\begin{array}{lll} \text{maximize} & f_1(x) + f_2(x) & \\ & & x_1 + 3x_2 \leq 12 \\ \text{subject to} & \text{at least two of the following} & 2x_1 + x_2 \leq 16 \\ & & x_1 + x_2 \leq 9 \\ & x_1, x_2 \geq 0 & \end{array}$$

where

$$f_1(x_1) = \begin{cases} 10 + 2x_1, & 0 \leq x_1 \leq 5 \\ 15 + x_1, & \text{otherwise} \end{cases}$$

$$f_2(x_2) = \begin{cases} 8 + x_2, & 0 \leq x_2 \leq 2 \\ 4 + 3x_2 & \text{otherwise} \end{cases}$$

Problem 7:

Solve the following problem using the branch-and-bound integer programming procedure describe in class. Feel free to do it graphically.

$$\begin{array}{ll} \text{maximize} & 2x_1 + x_2 \\ \text{subject to} & 2x_1 - 2x_2 \leq 3 \\ & -2x_1 + x_2 \leq 2 \\ & 2x_1 + 2x_2 \leq 13 \\ & x_1, x_2 \geq 0 \\ & x_1, x_2 \text{ integers} \end{array}$$

Problem 8:

Solve the following problem using implicit enumeration (the 0-1 programming technique) describe in class.

$$\begin{array}{ll} \text{minimize} & 3x_1 + 2x_2 + 5x_3 + x_4 \\ \text{subject to} & -2x_1 + x_2 - x_3 - 2x_4 \leq -2 \\ & -x_1 - 5x_2 - 2x_3 + 3x_4 \leq -3 \\ & x_1, x_2, x_3, x_4 \text{ binary} \end{array}$$

Implement **one** of the following algorithms:

1. Rupert's Meshing Algorithm
2. Garland and Heckbert's Surface Modeling Algorithm
3. Callahan and Kosaraju's Well-Separated Pair Decomposition Algorithm

Feel free to work in groups of two, although I will expect a better job from a group of two. Also feel free to come and discuss the algorithms with me. I've described each of these algorithms in class and have also made papers available that describe the algorithms in more detail. Two of the papers are available on the web, and the third is in a JACM issue that is available the CS reading room (Soda 681).

Note: Source code for some of these algorithms is available on the web. If you choose to do a problem, please do not look at anyone elses source until you have handed in your implementation. You should feel free, however, to use other peoples code for various subroutines such as implementing a heap. Make sure you reference the code.

I have made some code for incremental Delaunay Triangulation available, which you might find useful for either of the first two algorithms. This is taken from the book "Graphics Gems IV", edited by Paul Heckbert, Academic Press, 1994. A copy of the associated chapter is available outside my door. The code is also available as part of a compressed tar file associated with the book. You might find, however, that it is just as easy if not easier to design your code from scratch since deciphering the code is not trivial. Also, as I mentioned in class, you might find a triangle based representation simpler (the code uses the quadedge representation). If you do use any part of the code, make sure to note in your code which parts you used.

You might also find Jonathan Shewchuk's Show-me code (showme.c) useful for plotting a set of triangles, or boundaries.

Rupert's Meshing Algorithm

You should implement the algorithm and run it on the test sets given below and report back the number of triangles for minimum angles in the ranges specified. Ideally you should generate a postscript drawing of the final triangulations. Although the drawings are not required they will certainly help you debug your program. The algorirthm is described in:

> Jim Rupert. "A Delaunay Refinement Algorithm for Quality 2-Dimensional Mesh Generation." Journal of Algorithms, 18(3):548-585, May 1995.

The test data files: (Note: these use the Poly format from Jonathan Schewchuk's Triangle code)

- The letter A from Rupert's paper. Run this from 15 to 25 degrees in steps of 2 degrees.
- Lake Superior from Rupert's paper. Run this from 10 to 15 degrees in 1 degree steps. Note that this input has an internal angle that is just slightly larger than 15 degrees, so it will not work with larger angles.
- A guitar from Jonathan Schewchuk. Run this from 15 to 25 degrees in steps of 2 degrees.

Note that all data is included in a bounding box. This simplifies the problem of dealing with exterior triangles (all triangles in the box are considered interior). In your implementation you don't need to remove the triangles in the exterior (between box and surface) nor in the "holes". You can therefore ignore the "holes" data at the end of the .poly files. If you do chose to remove the triangles, however, you can use the "holes" data by starting at each triangle with the hole point and removing triangles outwards until you hit a boundary segment. If you choose to remove them please state this in your results.

Garland and Heckbert's Surface Modeling Algorithm

You should implement the algorithm and run it on the test sets given below and report back the number of triangles for a range of maximum errors. Again generating a drawing of the triangulation will be helpful but is not necessary. The algorirthm is described in:

> Michael Garland and Paul Heckbert. "Fast Polygonal Approximation of Terrains and Height Fields." CMU-CS-95-181, CS Dept, Carnegie Mellon U., Sept. 1995.

The algorithm I want you to implement is number III in the paper (the version I described in class).

The test data files: (Note: these are binary files in STM format)

- Crater lake
- Ashby Gap, Virginia
- Desert around Mt. Tiefort, California
- Ozark, Missouri

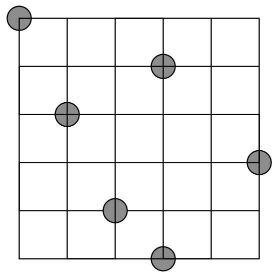
Callahan and Kosaraju's Well-Separated Pair Decomposition Algorithm:

You should implement the algorithm in 2 or 3 dimensions and run it on the Kuzmin model with 10K and 100K points (I originally said Plummer model, but I cannot find an equation for generating a Plummer distribution...if you do, please feel free to use it, but tell me.) For both sizes you should report the number of interactions as a function of the separation constant s ranging from 1 to 2 in increments of .2. The algorirthm is described in:

> Paul B. Callahan and S. Rao Kosaraju. "A Decomposition of Multi-Dimensional Point-Sets with Applications to k-Nearest-Neighbors and n-Body Potential Fields." JACM, 42(1):67-90, January 1995.
> Available for copying in the CS reading room.

Please feel free to send me email if there are any ambiguous questions.

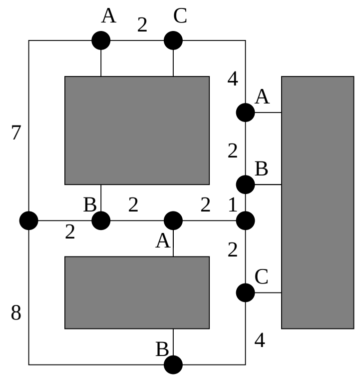
Problem 1: Consider the following graph.



A. Find a minimum cost Rectilinear Steiner Tree (RST) for the following set of points (the shaded points are the demand points).

B. Is this an S-RST? If not show the minimum cost S-RST.

Problem 2: For the following routing problem the three nets A, B and C need to be connected. The numbers on the edges represent their lengths, and assume that the capacity of every edge is 2.



A. Specify an integer (0-1) program for which the solution gives the minimum total wire length.

B. Solve the program (you do not need to show your work).

Problem 3: Describe an $O(nm)$ time $O(n+m)$ space algorithm for finding a minimum cost alignment assuming an affine gap cost model. Pseudocode with comments would be the best way to describe it.

Problem 4: (**Extra Credit.**) Describe an $O(nm\log(m+n))$ time algorithms for finding a minimum cost alignment assuming the gap cost function $x(k)$ (where k is the length of the gap) is concave downward (*i.e.* the second derivative is always negative).