CS 740, Fall 1996
Homework Assignment 5

Assigned: Nov. 12
Due: Thurs., Nov. 21, 1:00PM

1. Logistics

You may work in a group of up to 3 people in solving the problems for this assignment. You should turn in a single report for your entire group, identifying all of the group members.

Any clarifications and revisions to the assignment will be posted on Web page `assigns.html` in the class WWW directory.

You will hand in a hard copy document for this assignment. Formatted text is preferred to hand written.

2. Multimedia Computing

With all the craze about "Multimedia Computing," it is instructive to analyze the system requirements for processing video and audio data streams. Multimedia applications stretch the limits of all aspects of current computer technology, including: processor speeds, memory capacities, I/O throughput, and storage capacities.

We can identify the following tasks in a multimedia computing environment:

Capture An analog television signal is digitized into two tracks: one for video and one for audio, perhaps performing compression on the video stream. This needs to occur in real time.

Editting A series of video and audio clips are combined, and image processing is applied to perform special effects. This does not need to occur in real time.

Movie Generation The resulting video/audio sequence is converted into a movie, using further image compression and stored on CDROM or similar medium. This does not need to occur in real time.

Playback The movie is played on the consumer's computer from the CDROM. This must occur in real time.

We will focus on those steps requiring real-time processing.

2.1. Amateur Capture

The directory `/afs/cs.cmu.edu/user/bryant/asst5-pix` contains a number of Audio/Video clips stored in Quicktime format. You can view these clips on a Macintosh with the Quicktime Movie Player, or on a Unix workstation with the program `xanim`. This directory also includes documentation on `xanim` (in the files `xanim.readme`, `Examples.doc`) as well as a subdirectory containing the source code from `xanim` version 2.7.063. [**Note:** Be careful with these files. They are huge!].

The files named `dc`*XX*`.mov` were captured on a (relatively) low-end system consisting of a 75MHZ PowerPC Macintosh and a $200 video input card. All are clips of roughly the same 40 second section from the TV show "Davy Crockett." For a given screen resolution, the capture facility sets a frame rate based on an estimate of sustainable throughput. It captures the raw audio/video to disk, and then optionally applies software video compression to the video track. The video was captured at two resolutions: 120×160 (`dcs`*X*`.mov`) and 240×320(`dcb`*X*`.mov`). For the smaller resolution, the capture was able to sustain 8 frames per second, while for the larger only 4.5 frames per second were captured. All of the images are in color, with a color depth of 16 bits per pixel. The software compression algorithm required around 0.5 seconds for each frame of the larger images.

The files `dcsu.mov` and `dcbu.mov` have no video compression. [Check out their sizes!] The files `dcbs.mov` and `dcbv.mov` have "somewhat" and "very much" image compression applied, respectively. This image compression exploits only the spatial correlation in the image—each frame is compressed separately. The compressor uses a "lossy" compression technique, i.e., the resulting image is somewhat degraded in quality.

Note that all of these files include both video and audio tracks. The file `dcsu-snd.mov` contains just the audio track from the file `dcsu.mov`.

Task 1.A: Audio Storage Requirements and Compression Levels

Determine for each of the files `dc`*XX*`.mov`, *except for* `dcbt.mov` the following:

1. The audio sample rate and the number of bytes per sample.
2. The total running time.
3. The number of bytes required for the audio track.

In addition, determine how much one could gain by applying lossless compression, e.g., with the Unix program `compress` to the soundtrack. Use `dcsu-snd.mov` as your benchmark.

You can find this information by running `xanim` with the flags '`+v`' and '`+t`' set (as described in the `xanim` documentation files).

Task 1.B: Video Storage Requirements and Compression Levels

Determine for each of the files `dc`*XX*`.mov`, *except for* `dcbt.mov` the following:

1. The total number of frames, and the total number of pixels.
2. The number of bytes required to store the uncompressed image.
3. The number of bytes used to store the compressed image.

4. The video compression ratio, i.e., the ratio between the compressed and uncompressed sizes.
5. How much further compression could be achieved using lossless compression, e.g., using the Unix program `compress`.

Task 1.C: Image/Sound Quality

Compare the image and sound qualities of the files dc*XX*.mov, *except for* `dcbt.mov`. Which of the following artifacts can you observe, and to what degree. Note that some of these artifacts are a function of the system on which you display the clips, while others are a function of the capturing.

- Jerkiness due to a low frame rate.
- Granularity due to a low resolution.
- Loss of detail due to compression.
- Color aliasing due to limited depth in your system's graphics system.
- Poor sound quality due to a low sampling rate.
- Poor sound quality due to the limited throughput of the capture system.

Task 1.D: Amateur Bandwidth/Processing Requirements

Determine for each of the files dc*XX*.mov, *except for* `dcbt.mov`, the following:

1. The bandwidth required by the raw video capture.
2. The bandwidth required by the audio capture.
3. The bandwidth required to display the compressed images.

Determine the processing requirements to perform software compression, expressed in clock cycles per pixel. Use the compression times observed for files `dcbs.mov` and `dcbv.mov` as your benchmark.

2.2. Professional Capture

As you have observed, the videos captured by this low-end system do not come close to the quality of broadcast television. For working with "NTSC" quality videos, the accepted standard is to digitize at 480×640 pixels at a rate of 30 frames per second, with a color depth of 24 bits. In addition, we would like "CD" quality sound, consisting of two stereo channels, sampled at 38,400 samples per second, with 2 bytes per sample.

Task 1.E: Professional Bandwidth/Processing Requirements

Compute estimates of the following:

1. The bandwidth required for the raw video stream.
2. The bandwidth required for a compressed video stream, assuming the compression ratio achieved by the file `dcbs.mov`.
3. The bandwidth required for the audio stream.
4. The processing requirement required to perform real-time compression in software, assuming the algorithm scales linearly with the pixel rate.
5. Which would be more cost effective: upgrading to a workstation with this processing capability, or to buy a video capture board performing this level of compression in hardware for $2000?

Task 1.F: Disk Requirements

Consider the following storage devices, both available commercially:

- A 4.3 GB disk with a sustained transfer rate of 4.8 MB/second, costing $1000.
- A 16 GB RAID system with a sustained transfer rate of 30 MB/second, costing $6000.

Raw audio and video streams have uniform bandwidth requirements. A compressed video stream is somewhat less uniform—some frames can be compressed more than others. A rule of thumb for compressed video is that the hardware must be able to sustain $1.5\times$ the nominal bandwidth.

1. How many minutes of uncompressed video+audio could be stored on each?
2. How many minutes of compressed video+audio could be stored on each?
3. Which could keep up with the raw capture rates?
4. Which could keep up with the rates for the compressed streams?

2.3. Video Playback

Once the clips have been assembled the final movie can be generated. In this step, further image compression is performed that exploits the temporal correlation between successive frames in the image. The clip in the file `dcbt.mov` was obtained from `dcbv.mov` using this technique. Note that the space reduction was small, due to the low frame rate of this clip. With a higher frame rate, the difference between successive images is lower. Temporal compression involves transmitting two types of frames: "key" frames containing the complete frame image, and "delta" frames describing local updates of the previous frame.

Task 1.G: Playback Image Quality

What artifacts of the temporal compression can be observed in the file `dcbt.mov`?

The files `babe5.mov` and `babe10.mov` were obtained over the Internet. They are previews from the movie "Babe," digitized at 5 and 10 frames per second, respectively. Their resolution and frame rates are low to reduce the total file size. Nonetheless, they are representative of what can be obtained by more a more sophisticated capture and compression system.

1. Determine the sizes of the raw video and audio tracks.
2. Determine the compression ratio for the video tracks.
3. What artifacts can you observe that indicate that temporal compression has been applied to the video tracks?
4. In what way is the temporal video compression superior to what was performed to generate the file `dcbt.mov`?

Task 1.H: CDROM/DVD Technology

The new MPEG-2 video compression standard is designed to allow NTSC-quality videos to be played from low cost media such as CDROMs. They claim that images can be compressed to such a degree that they require a bandwidth of just 440 KB/second for playback. In addition, audio would be played in the "CD-quality" format described earlier.

An industry consortium is currently developing standards for a successor to the CDROM known as a "Digital Video Disk" (DVD). Each DVD will have two sides, with a storage capacity of 4.7 GB per side.

1. What compression ratio does this bandwidth represent over the raw NTSC-quality video?
2. What class of CDROM drive (measured relative to an audio CD player) would be required to provide this bandwidth?
3. How many minutes of video+audio of this quality could be stored on a standard CDROM?
4. How many minutes of video+audio of this quality could be stored on a DVD?

Task 1.I: Playback Processing Requirements

Suppose you connected a CDROM or DVD drive of appropriate bandwidth and capacity to your current system. Do you think your processor would be capable of displaying NTSC-quality video in real time, assuming all decompression is performed in software? To get a feeling for the amount of processing required, measure the CPU and memory utilization of `xanim` while displaying both compressed and uncompressed clips. [One way to do this is to run the Unix program `top` in a separate window while `xanim` is running]. Make rough estimates of how these requirements would scale for higher quality video.

3. Rearrangeable networks

This series of problems studies a class of networks called *rearrangeable* networks. Like the networks that we discussed in class, a rearrangeable network is composed of switches and edges. For the purposes of this problem, each switch has two incoming edges i_1 and i_2 and two outgoing edges o_1 and o_2. A switch has two possible states. In the "straight-through" state, it connects i_1 to o_1 and i_2 to o_2. In the "exchange" state, it connects i_1 to o_2 and i_2 to o_1.

A rearrangeable network has n special edges called its *input terminals* (or just inputs) and another n special edges called its *output terminals* (or just outputs). The input terminals are labeled 0 through $n - 1$, as are the output terminals. The input terminals serve as the incoming edges to a special set of $n/2$ switches (called the input switches), but they do not serve as the outgoing edges of any switches. Similarly, the output terminals serve as the outgoing edges of set of $n/2$ switches called the output switches.

A rearrangeable network routes paths from the input terminals to the output terminals. Because each switch can only permute the connection between its incoming edges and outgoing edges (and not connect both incoming edges to the same outgoing edge, or both outgoing edges to the same incoming edge), each input terminal can be connected by a path to only one output terminal, and vice versa. Thus, the connections between the input terminals and output terminals can be described by a one-to-one mapping. Since the input and output terminals are both labeled 0 through $n - 1$, this mapping is also a partial (or full) permutation of the numbers 0 through $n - 1$.

In order for a network to qualify as a rearrangeable network, it must be able connect its input terminals to its output terminals according to any permutation.

Figure 1 shows a rearrangeable network called a Beneš network. The network in this figure has 8 input terminals, numbered 0 through 7, and 8 output terminals, also numbered 0 through 7. The input terminals are connected to the output terminals according to the following permutation ϕ: $\phi(0) = 3$, $\phi(1) = 1$, $\phi(2) = 2$, $\phi(3) = 6$, $\phi(4) = 0$, $\phi(5) = 5$, $\phi(6) = 4$, $\phi(7) = 7$.

3.1. Task 2.A: A lower bound on the size of any rearrangeable network

In this part of the assignment, you are going to prove that any rearrangeable network with n input terminals and n output terminals must have at least $\Omega(n \log n)$ switches. This lower bound is one of the first theorems proved about interconnection networks. (It was proved by Shannon in 1950.)

Here is a big hint. You should relate the number of different permutations to the number of different possible states that the network can be in. Since each switch can take on only two different states, it is not hard to count the number of states that the network can be in.

The Ω notation hides a leading constant. Can you estimate the constant implied by your bound?

3.2. Task 2.B: Deriving the Beneš network

The reason that the Beneš network is rearrangeable may seem mysterious at first, but it turns out that its structure is fairly easy to derive. The key is to view the network as consisting of three parts. The first part is the set of $n/2$ input switches. The second part is a pair of recursively-constructed

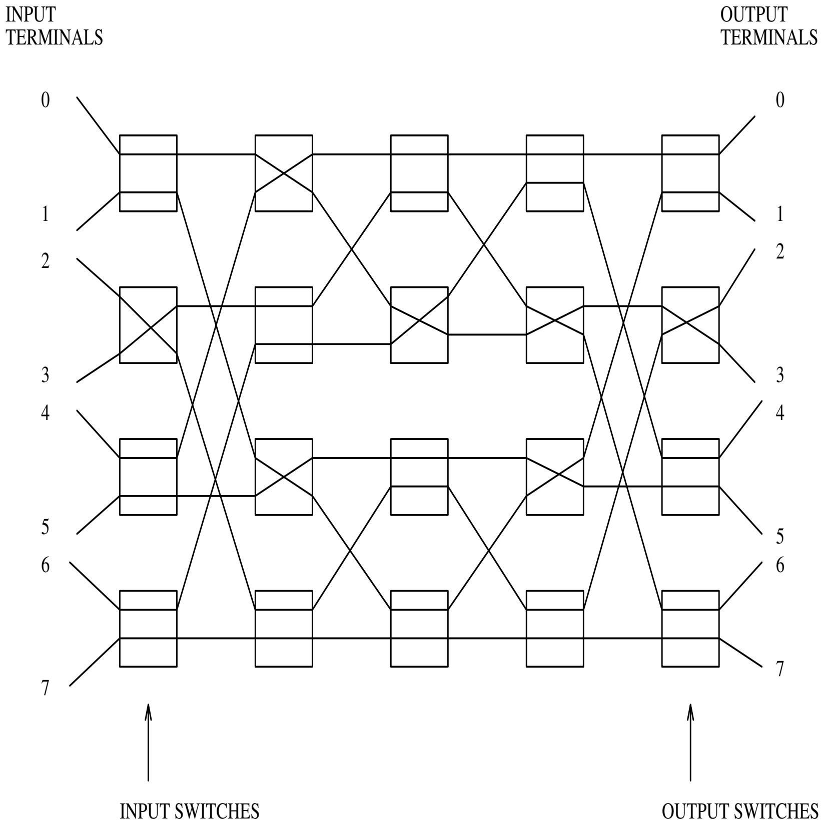

Figure 1: A Beneš network with 8 input terminals.

Beneš networks, each with $n/4$ input and output switches. Let's call these the top and bottom sub-Beneš networks. The third part is the set of $n/2$ output switches. One of the edges leading out of each input switch in the first part is connected to an input switch of the top sub-Beneš network, and the other is connected to an input switch of the bottom sub-Beneš network. Similarly, each output switch in the third part is connected to output switches in both the top and botton sub-Beneš networks. The base case for the recursion is the Beneš network that consists of just a single switch, which only has to permute the numbers 0 and 1.

Prove that the Beneš network is rearrangeable.

Here are two big hints. First, your proof should be by induction. In particular, you should assume that the sub-Beneš networks are rearrangeable. Second, use the row-column-row permuting theorem that was proved in class. Your array should have $n/2$ rows, and 2 columns. The rows will correspond to the input and output switches, and the columns will correspond to the sub-Beneš networks.

3.3. Task 2.C: Building a rearrangeable network out of butterfly networks

In this part of the assignment, we will consider how many copies of the butterfly network are needed to construct a rearrangeable network. In other words, suppose that we were to connect several butterfly networks in series, so that the ith output switch of one butterfly serves as the ith input switch of the next butterfly. (Note that the Beneš network is *not* constructed this way – it consists of two back-to-back butterflies, rather than two butterflies in series.)

How many butterflies are required to build a rearrangeable network? Start off by proving that one butterfly is not enough. After solving the problem in Part A, this should not be too difficult for you to do.

Now show that 4 butterflies are sufficient. There is a trick to doing this. We are going to make the 4 butterflies simulate a Beneš network. The direction of one of the two butterflies is reversed in the Beneš network. We're not allowed to reverse the direction of any of our butterflies, but suppose that we were able to build a black box with n input terminals and n output terminals that could permute the n paths that entered it so that the next butterfly that they entered appeared to them to be reversed. The permutation that will do this is called the *bit-reversal permutation*. This permutation is defined as follows. Suppose that we view the labels of the input and output terminals as $\log_2 n$-bit binary numbers. Then the input terminal labed $b_{(\log_2 n)-1}, \ldots, b_1, b_0$ should be connected to the output terminal labeled $b_0, b_1, \ldots, b_{(\log_2 n)-1}$.

1. Prove that a butterfly, followed by a black box implementing the bit-reversal permutation, followed by another butterfly can perform any permutation that the Beneš network can perform.
2. Show that, for $n \geq 8$, the bit-reversal permutation can not be implemented by one butterfly.
3. Now show that the bit-reversal permutation can be implemented by two butterflies.

Here are some hints that will help with the third item.

In solving this problem it helps to add labels to the internal edges of the butterfly as well as the input and output terminals. We will view each edge label as a $(\log n)$-bit binary number. At each level, the edges are assigned labels based on where they enter the switches at that level. The labels run consecutively from the top to bottom of the network.

In the first butterfly, suppose that a path uses an edge labeled

$$c_{(\log_2 n)-1}, \ldots, c_i, c_{i-1}, \ldots, c_1, c_0$$

to enter a switch on level i (where $(\log_2 n) - 1 \geq i \geq 0$). Then it should enter a switch on level $i + 1$ using the edge labeled

$$c_{(\log_2 n)-1}, \ldots, c_i \oplus c_{(\log_2 n)-1-i}, c_{i-1}, \ldots, c_1, c_0,$$

where $\oplus$ is the exclusive-OR operator. Show that this rule prevents any two paths from using the same edge. Then derive the rule for routing the paths in the second butterfly.

With a lot more effort, it's possible to prove that 3 butterflies are sufficient too. (You don't have to do this!) No one knows if 2 butterflies are sufficient.