Combinatorial algorithms

computing subset rank and unrank, Gray codes,
computing permutation rank and unrank
computing graph isomorphism

Jiří Vyskočil, Radek Mařík

2011

Combinatorial Generation

- **definition:**

Suppose that S is a finite set. A *ranking function* will be a bijection

$$\text{rank}: S \rightarrow \{0, \ldots, |S|\text{-}1\}$$

and unrank function is an inverse function to rank function.

- **definition:**

Given a ranking function rank , defined on S, the successor function satisfies the following rule:

$$\text{successor}(s) = t \quad \Leftrightarrow \quad \text{rank}(t) = \text{rank}(s) + 1$$

- **potential uses:**
  - storing combinatorial objects in the computer instead of storing a combinatorial structure which could be quite complicated
  - generation of random object from S ensuring equal probability $1/|S|$

Subsets

- **computing the subset rank over lexicographical ordering**

```
1) Function SubsetLexRank( size n; set T ) : rank
2) r = 0 ;
3) for i = 1 to n do {
4)     if i ∈ T then r = r + 2^(n-i) ;
5) }
6) return r
```

```
1) Function SubsetLexUnrank( size n; rank r ) : set
2) T = ∅ ;
3) for i = n downto 1 do {
4)     if r mod 2 = 1 then T = T ∪ {i} ;
5)     r = ⌊ r/2 ⌋ ;
6) }
7) return T
```

Gray Code

- **definition:**

The *reflected binary code*, also known as *Gray code*, is a binary numeral system where two successive values differ in only one bit.

G^n will denote the reflected binary code for 2^n binary n-tuples, and it will be written as a list of 2^n vectors, as follows:

$$G^n=[G_0^n, G_1^n, \dots, G_{2^n-1}^n]$$

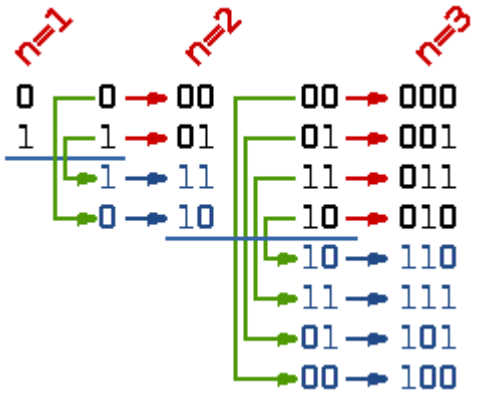
The codes G^n are defined recursively:

$$G^1=[0,1]$$

$$G^n=[0G_0^{n-1}, 0G_1^{n-1}, \dots, 0G_{2^n-1}^{n-1}, 1G_{2^n-1}^{n-1}, \dots, 1G_1^{n-1}, 1G_0^{n-1}]$$

- **example:**

$G^3=[000, 001, 011, 010, 110, 111, 101, 100]$



Gray Code

- **lemma 1**

Suppose

- $0 \le r \le 2^n - 1$
- $B = b_n, \ldots, b_0$ is a binary code of r
- $G = g_n, \ldots, g_0$ is a Gray code of r

Then for every $j \in \{0,1, \ldots, n-1\}$

$$g_j = (b_j + b_{j+1}) \bmod 2$$

- **proof**

By induction on n.

Gray Code

- **lemma 2**

Suppose

- $0 \le r \le 2^n - 1$
- $B = b_n, \ldots, b_0$ is a binary code of r
- $G = g_n, \ldots, g_0$ is a Gray code of r

Then for every $j \in \{0,1, \ldots, n-1\}$

$$b_j = (g_j + b_{j+1}) \bmod 2$$

- **proof**

$$g_j = (b_j + b_{j+1}) \bmod 2 \Rightarrow g_j \equiv (b_j + b_{j+1}) \pmod 2 \Rightarrow$$
$$b_j \equiv (g_j + b_{j+1}) \pmod 2 \Rightarrow b_j = (g_j + b_{j+1}) \bmod 2$$

Gray Code

- **lemma 3**

Suppose

- $0 \le r \le 2^n - 1$
- $B = b_n, \ldots, b_0$ is a binary code of r
- $G = g_n, \ldots, g_0$ is a Gray code of r

Then for every $j \in \{0,1, \ldots, n-1\}$

$$b_j = \left(\sum_{i=j}^{n-1} g_i \right) \bmod 2$$

- **proof**

$$\left(\sum_{i=j}^{n-1} g_i \right) \bmod 2 = \left(\sum_{i=j}^{n-1} (b_i + b_{i+1}) \right) \bmod 2 = \left(b_j + b_n + 2 \sum_{i=j+1}^{n-1} b_i \right) \bmod 2 = (b_j + b_n) \bmod 2 = b_j$$

By lemma 2.

By the sum reordering.

By the property of modulo.

By the maximum range of r and the range of b_j.

Gray Code

- **converting to and from minimal change ordering (Gray code)**

```
1) Function BinaryToGray( binary code rank B ) : gray code rank
2) return B xor (B >> 1)
```

```
1) Function GrayToBinary(gray code rank G ) : binary code rank
2) B = 0;
3) n = (number of bits in G) – 1;
4)  for i=0 to n do {
5)    B = B << 1;
6)    B = B or (1 and ( (B >> 1) xor (G >> n) ) );
7)    G = G << 1;
8)  }
9)  return B
```

Subsets – Gray Code

- **computing the subset rank over minimal change ordering**

```
1) Function GrayCodeRank( size n; set T ) : rank
2) r = 0 ;
3) b = 0 ;
4) for i = n – 1 downto 0 do {
5)     if n – i ∈ T then b = 1 – b ;
6)     if b = 1 then r = r + 2^i ;
7) }
8) return r
```

Subsets – Gray Code

- **computing the subset unrank over minimal change ordering**

1) **Function** GRAYCODEUNRANK(size n; rank r) : set
2) $T = \emptyset$;
3) $c = 0$;
4) **for** $i = n - 1$ **downto** 0 **do** **{**
5) $b = \left\lfloor \frac{r}{2^i} \right\rfloor$;
6) **if** $b \neq c$ **then** $T = T \cup \{n - i\}$;
7) $c = b$;
8) $r = r - b \cdot 2^i$;
9) **}**
10) **return** T

Permutations

- **computing the permutation rank over lexicographical ordering**

```
1)  Function PermLexRank( size n; permutation π ) : rank
2)  r = 0 ;
3)  ρ = π ;
4)  for j = 1 to n do {
5)      r = r + (ρ[ j ] – 1)·(n – j)! ;
6)      for i = j +1 to n do if ρ[ i ] > ρ[ j ] then ρ[ i ] = ρ[ i ] – 1 ;
7)  }
8)  return r
```

Permutations

- **computing the permutation unrank over lexicographical ordering**

1) **Function** PERMLEXUNRANK(size n; rank r) : permutation
2) $\pi[\, n \,] = 1$;
3) **for** $j = 1$ **to** $n - 1$ **do** {
4) $\quad d = \dfrac{r \bmod (j+1)!}{j!}$;
5) $\quad r = r - d \cdot j!$;
6) $\quad \pi[\, n - j \,] = d + 1$;
7) $\quad$ **for** $i = n - j + 1$ **to** n **do** **if** $\pi[\, i \,] > d$ **then** $\pi[\, i \,] = \pi[\, i \,] + 1$;
8) }
9) **return** π

Computing Graph Isomorphism

- **definition:**

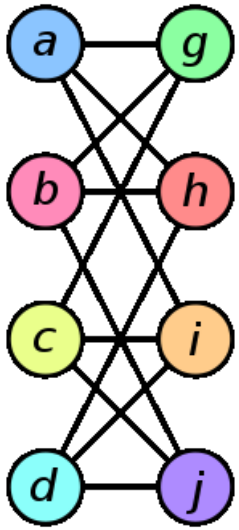
Two graphs $G_1=(V_1,E_1)$ and $G_2=(V_2,E_2)$ are *isomorphic* if there is a bijection $f: V_1 \to V_2$ such that

$$\forall\, x,y \in V_1 \quad : \quad \{f(x),f(y)\} \in E_2 \quad \Leftrightarrow \quad \{x,y\} \in E_1$$

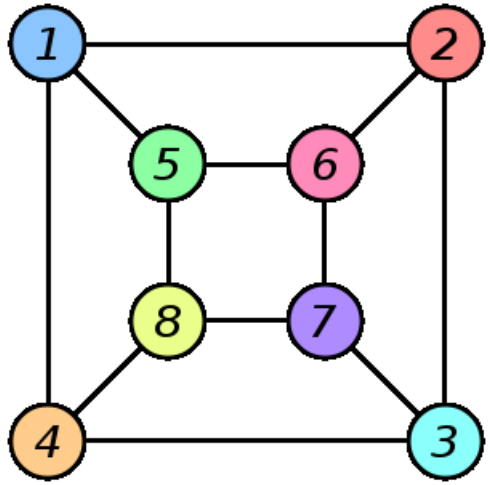
The mapping f is said to be an *isomorphism* between G_1 and G_2.

- **example:**

G_1 :



G_2 :



f :

$f(a) = 1$
$f(b) = 6$
$f(c) = 8$
$f(d) = 3$
$f(g) = 5$
$f(h) = 2$
$f(i) = 4$
$f(j) = 7$

Computing Graph Isomorphism

- **definition of invariant:**

Let $\mathcal{F}$ be a family of graphs. An *invariant* on $\mathcal{F}$ is a function Φ with domain $\mathcal{F}$ such that

$$\forall\, G_1, G_2 \in \mathcal{F} \;:\; \Phi(G_1) = \Phi(G_2) \;\Leftarrow\; G_1 \text{ is isomorphic to } G_2$$

- **example:**
  - $|V|$ for graph $G=(V, E)$ is an invariant.
  - The following degree sequence $[\deg(v_1), \deg(v_2), \deg(v_3), \ldots, \deg(v_n)]$ is not an invariant.
  - However, if the degree sequence is sorted in non-decreasing order, then it is an invariant.

Computing Graph Isomorphism

- **definition :**

Let $\mathcal{F}$ be a family of graphs on vertex set V and let D be a function with domain ($\mathcal{F} \times V$). Then the *partition induced* by D is

$$B = [\ |B[0]|, |B[1]|, \ldots, |B[n-1]|\ ]$$

where

$$B[i] = \{\ v \in V\ :\ D\ (G,v) = i\ \}$$

If the function

$$\Phi_D(G) = [\ |B[0]|, |B[1]|, \ldots, |B[n-1]|\ ]$$

is an invariant, then we say that D is an *invariant inducing function*.

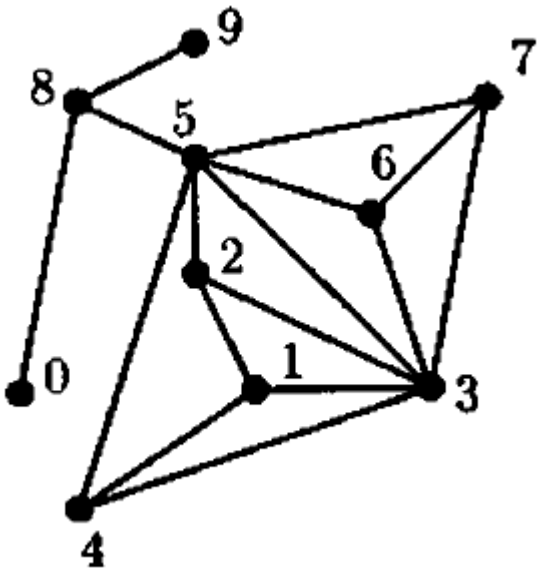
Computing Graph Isomorphism - Example

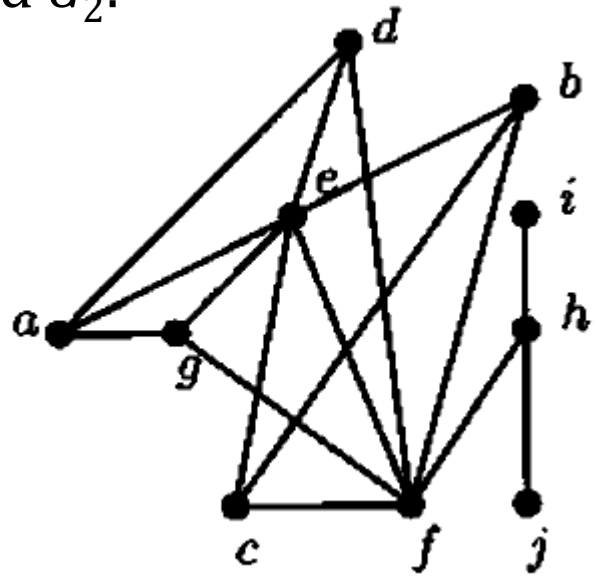
Let

- $D_1(G,x)=\deg_G(x)$
- $D_2(G,x)=[d_j(x) : j = 1,2, \ldots , d_{n-1}]$

  where $d_j(x)=|\{y : y \text{ is adjacent to } x \text{ and } \deg_G(y) = j \}|$

Suppose the following graphs G_1 and G_2:





Computing Graph Isomorphism - Example

$$X_0(\mathcal{G}_1) = \{0,1,2,3,4,5,6,7,8,9\}.$$

$$X_0(\mathcal{G}_2) = \{a,b,c,d,e,f,g,h,i,j\}.$$

x	0	1	2	3	4	5	6	7	8	9
$D_1(\mathcal{G}_1, x)$	1	3	3	6	3	6	3	3	3	1

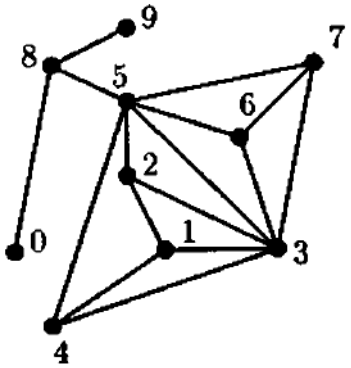
$$\Downarrow$$

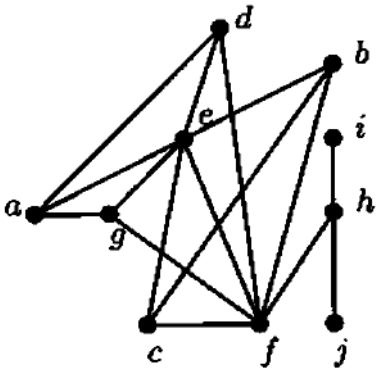
$$X_1(\mathcal{G}_1) = \{0,9\},\{1,2,4,6,7,8\},\{3,5\}$$

$\bar{x}$	a	b	c	d	e	f	g	h	i	j
$D_1(\mathcal{G}_2, \bar{x})$	3	3	3	3	6	6	3	3	1	1

$$\Downarrow$$

$$X_1(\mathcal{G}_2) = \{i,j\},\{a,b,c,d,g,h\},\{e,f\}.$$



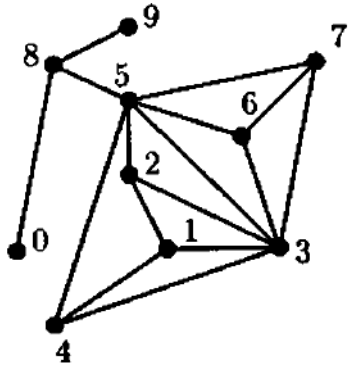


Computing Graph Isomorphism - Example

$$D_2(\mathcal{G}_1, 0) = (0,0,1,0,0,0,0,0,0)$$
$$D_2(\mathcal{G}_1, 1) = (0,0,2,0,0,1,0,0,0)$$
$$D_2(\mathcal{G}_1, 2) = (0,0,1,0,0,2,0,0,0)$$
$$D_2(\mathcal{G}_1, 3) = (0,0,5,0,0,1,0,0,0)$$
$$D_2(\mathcal{G}_1, 4) = (0,0,1,0,0,2,0,0,0)$$
$$D_2(\mathcal{G}_1, 5) = (0,0,5,0,0,1,0,0,0)$$
$$D_2(\mathcal{G}_1, 6) = (0,0,1,0,0,2,0,0,0)$$
$$D_2(\mathcal{G}_1, 7) = (0,0,1,0,0,2,0,0,0)$$
$$D_2(\mathcal{G}_1, 8) = (2,0,0,0,0,1,0,0,0)$$
$$D_2(\mathcal{G}_1, 9) = (0,0,1,0,0,0,0,0,0)$$

$$\Downarrow$$

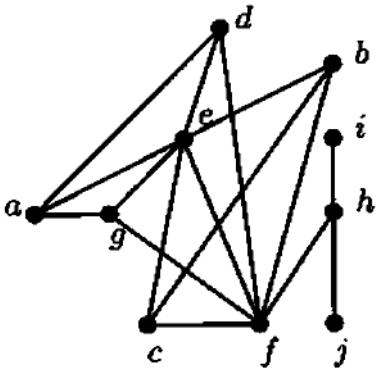
$$X_2(\mathcal{G}_1) = \{0,9\}, \{8\}, \{2,4,6,7\}, \{1\}, \{3,5\}.$$



$$D_2(\mathcal{G}_2, a) = (0,0,2,0,0,1,0,0,0)$$
$$D_2(\mathcal{G}_2, b) = (0,0,1,0,0,2,0,0,0)$$
$$D_2(\mathcal{G}_2, c) = (0,0,1,0,0,2,0,0,0)$$
$$D_2(\mathcal{G}_2, d) = (0,0,1,0,0,2,0,0,0)$$
$$D_2(\mathcal{G}_2, e) = (0,0,5,0,0,1,0,0,0)$$
$$D_2(\mathcal{G}_2, f) = (0,0,5,0,0,1,0,0,0)$$
$$D_2(\mathcal{G}_2, g) = (0,0,1,0,0,2,0,0,0)$$
$$D_2(\mathcal{G}_2, h) = (2,0,0,0,0,1,0,0,0)$$
$$D_2(\mathcal{G}_2, i) = (0,0,1,0,0,0,0,0,0)$$
$$D_2(\mathcal{G}_2, j) = (0,0,1,0,0,1,0,0,0)$$

$$\Downarrow$$

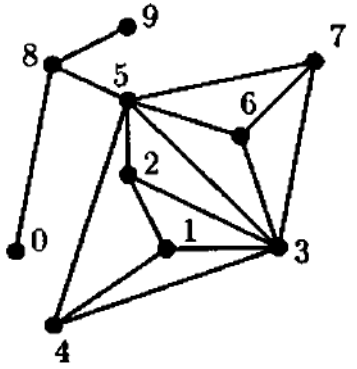
$$X_2(\mathcal{G}_2) = \{i,j\}, \{h\}, \{b,c,d,g\}, \{a\}, \{e,f\}.$$



Computing Graph Isomorphism - Example

This restricts a possible isomorphism to bijections between the following sets:

$$
\begin{aligned}
\{0,9\} &\longleftrightarrow \{i,j\} \\
\{8\} &\longleftrightarrow \{h\} \\
\{2,4,6,7\} &\longleftrightarrow \{b,c,d,g\} \\
\{1\} &\longleftrightarrow \{a\} \\
\{3,5\} &\longleftrightarrow \{e,f\}
\end{aligned}
$$



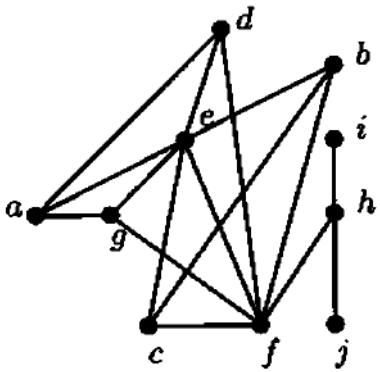
There are $96 = (2!)(1!)(4!)(1!)(2!)$ bijections giving the possible isomorphisms. Examination of each of these possible isomorphisms shows that only the following eight bijections are isomorphisms.

$$\begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ i & a & d & e & g & f & c & b & h & j \end{pmatrix} \qquad \begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ j & a & d & e & g & f & c & b & h & i \end{pmatrix}$$

$$\begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ i & a & d & e & g & f & b & c & h & j \end{pmatrix} \qquad \begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ j & a & d & e & g & f & b & c & h & i \end{pmatrix}$$

$$\begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ i & a & e & d & g & f & c & b & h & j \end{pmatrix} \qquad \begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ j & a & e & d & g & f & c & b & h & i \end{pmatrix}$$

$$\begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ i & a & e & d & g & f & b & c & h & j \end{pmatrix} \qquad \begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ j & a & e & d & g & f & b & c & h & i \end{pmatrix}$$



Computing Graph Isomorphism

```
1)  Function FindIsomorphism (set of invariant inducing functions I; graph G1, G2) : set of isomorphisms
2)  try {
3)      ( N, X, Y ) = GetPartitions (I, G1, G2) ;
4)  }
5)  catch ("G1 and G2 are not isomorphic!") { return ∅ ; }
6)  for i = 0 to N – 1 do {
7)      for each x ∈ X [i] do {
8)          W [x] = i ;
9)      }
10) }
11) return CollectIsomorphisms(G1, G2, 0, Y, W, f)
```

Computing Graph Isomorphism

1) **Function** GETPARTITIONS $\begin{pmatrix}\text{set of invariant inducing functions } I;\\ \text{graph } G_1;\\ \text{graph } G_2\end{pmatrix} : \begin{pmatrix}\text{number of partitions},\\ \text{parititions of } G_1,\\ \text{parititions of } G_2\end{pmatrix}$
2) $X[0]$ = vertices of G_1; $Y[0]$ = vertices of G_2; $N = 1$;
3) **for each** $D \in I$ **do** {
4) $P = N$;
5) **for** $i = 0$ **to** $P - 1$ **do** {
6) Partition $X[i]$ into sets $X_1[i], X_2[i], X_3[i], \ldots, X_m[i]$ where $x,y \in X_j[i] \Leftrightarrow D(G_1,x) = D(G_1,y)$;
7) Partition $Y[i]$ into sets $Y_1[i], Y_2[i], Y_3[i], \ldots, Y_n[i]$ where $x,y \in Y_j[i] \Leftrightarrow D(G_2,x) = D(G_2,y)$;
8) **if** $n \neq m$ **then throw exception** "G_1 and G_2 are not isomorphic!";
9) Order $Y[i]$ into sets $Y_1[i], Y_2[i], Y_3[i], \ldots, Y_n[i]$ so that
10) $\forall x \in X[i], \forall y \in Y[i] : D(G_1,x) = D(G_2,y) \Leftrightarrow x \in X_j[i]$ and $y \in Y_j[i]$;
11) **if** ordering is not possible **then throw exception** "G_1 and G_2 are not isomorphic!";
12) $N = N + m - 1$;
13) }
14) Reorder the partitions so that: $|X[i]| = |Y[i]| \leq |X[i+1]| = |Y[i+1]|$ for $0 \leq i < N - 1$;
15) }
16) **return** (N, X, Y)

Computing Graph Isomorphism

```
1)  Function CollectIsomorphisms( graph G1, G2 ; partition mapping W as current isomorphism f as
                                  starting vertex of G1 v ; array [vertices of G1] of ; array [vertices of G1] of
                                  parititions of G2 Y ; indices of partitions of G1 vertices of G2 ) : set of isomorphisms
2)  if v = number of vertices of G1 then return { f } ;
3)  R = ∅ ;
4)  p = W [v] ;
5)  for each y ∈ Y [p] do {
6)      OK = true ;
7)      for u = 0 to v – 1 do {
8)          if ( ( { u ,v } ∈ edges of G1 and { f [u], y } ∉ edges of G2 )
                                          or
               ( { u ,v } ∉ edges of G1 and { f [u], y } ∈ edges of G2 ) ) then OK = false ;
9)      }
10)     if OK then {
11)         f [v] = y ;
12)         R = R ∪ CollectIsomorphisms(G1, G2, v+1, Y, W, f ) ;
13)     }
14) }
15) return R
```

References

- D.L. Kreher and D.R. Stinson , *Combinatorial Algorithms: Generation, Enumeration and Search* , CRC press LTC , Boca Raton, Florida, 1998.