Effiziente Programme

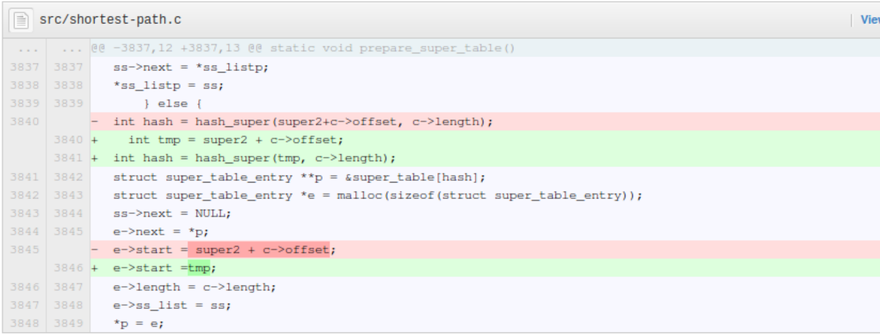
F. Gruber, P. Hönisch, C. Hochreiner, M. Petritsch

21. 1. 2011

Test Set-Up

- Prozessor: i7
- Testprogramm: papiex
- Testscript: testet gleichzeitig auf o0 und o3
- Versionskontrolle: git

Replaced computation with temp



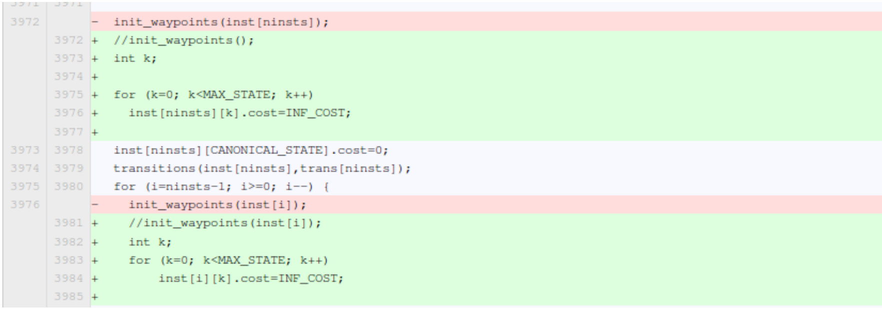
src/shortest-path.c

```
@@ -3837,12 +3837,13 @@ static void prepare_super_table()
   ss->next = *ss_listp;
   *ss_listp = ss;
       } else {
-  int hash = hash_super(super2+c->offset, c->length);
+    int tmp = super2 + c->offset;
+  int hash = hash_super(tmp, c->length);
   struct super_table_entry **p = &super_table[hash];
   struct super_table_entry *e = malloc(sizeof(struct super_table_entry));
   ss->next = NULL;
   e->next = *p;
-  e->start = super2 + c->offset;
+  e->start =tmp;
   e->length = c->length;
   e->ss_list = ss;
   *p = e;
```

Replaced computation with temp

Result

Inline method

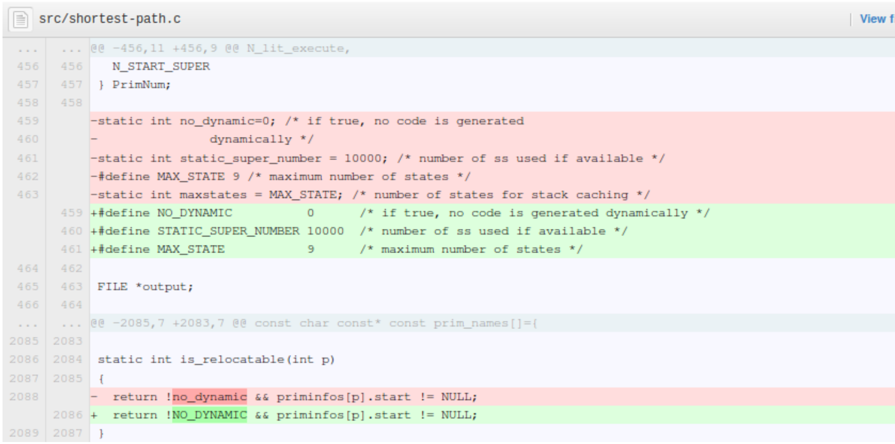


```
-  init_waypoints(inst[ninsts]);
+  //init_waypoints();
+  int k;
+
+  for (k=0; k<MAX_STATE; k++)
+    inst[ninsts][k].cost=INF_COST;
+
   inst[ninsts][CANONICAL_STATE].cost=0;
   transitions(inst[ninsts],trans[ninsts]);
   for (i=ninsts-1; i>=0; i--) {
-    init_waypoints(inst[i]);
+    //init_waypoints(inst[i]);
+    int k;
+    for (k=0; k<MAX_STATE; k++)
+        inst[i][k].cost=INF_COST;
+
```

Inline method

Result

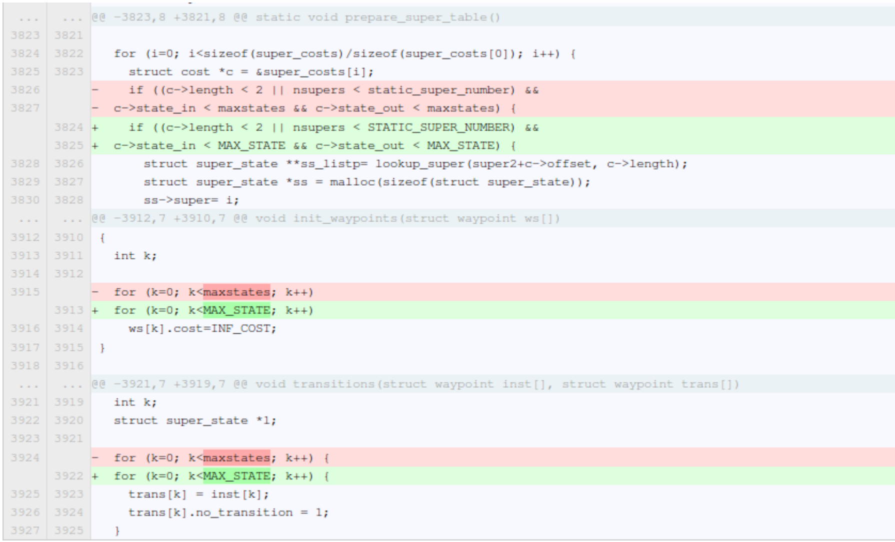
Make global constants makros



src/shortest-path.c

```
@@ -456,11 +456,9 @@ N_lit_execute,
   N_START_SUPER
 } PrimNum;

-static int no_dynamic=0; /* if true, no code is generated
-                 dynamically */
-static int static_super_number = 10000; /* number of ss used if available */
-#define MAX_STATE 9 /* maximum number of states */
-static int maxstates = MAX_STATE; /* number of states for stack caching */
+#define NO_DYNAMIC          0       /* if true, no code is generated dynamically */
+#define STATIC_SUPER_NUMBER 10000   /* number of ss used if available */
+#define MAX_STATE           9       /* maximum number of states */

 FILE *output;

@@ -2085,7 +2083,7 @@ const char const* const prim_names[]={

 static int is_relocatable(int p)
 {
-  return !no_dynamic && priminfos[p].start != NULL;
+  return !NO_DYNAMIC && priminfos[p].start != NULL;
 }
```

Make global constants makros

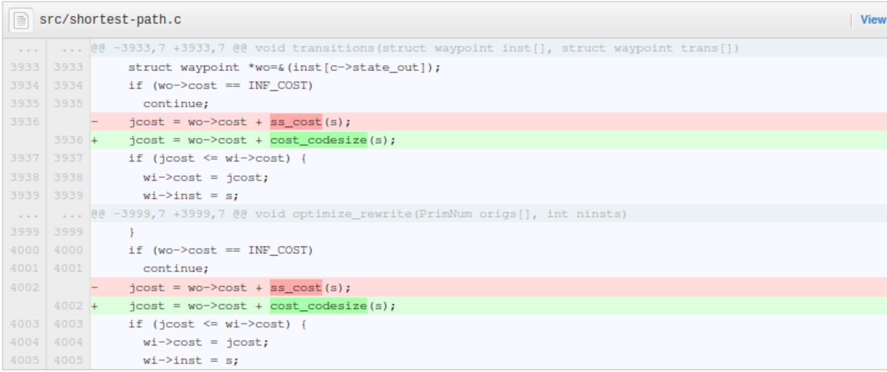


```
@@ -3823,8 +3821,8 @@ static void prepare_super_table()

   for (i=0; i<sizeof(super_costs)/sizeof(super_costs[0]); i++) {
     struct cost *c = &super_costs[i];
-    if ((c->length < 2 || nsupers < static_super_number) &&
-  c->state_in < maxstates && c->state_out < maxstates) {
+    if ((c->length < 2 || nsupers < STATIC_SUPER_NUMBER) &&
+  c->state_in < MAX_STATE && c->state_out < MAX_STATE) {
       struct super_state **ss_listp= lookup_super(super2+c->offset, c->length);
       struct super_state *ss = malloc(sizeof(struct super_state));
       ss->super= i;
@@ -3912,7 +3910,7 @@ void init_waypoints(struct waypoint ws[])
 {
   int k;

-  for (k=0; k<maxstates; k++)
+  for (k=0; k<MAX_STATE; k++)
     ws[k].cost=INF_COST;
 }

@@ -3921,7 +3919,7 @@ void transitions(struct waypoint inst[], struct waypoint trans[])
   int k;
   struct super_state *l;

-  for (k=0; k<maxstates; k++) {
+  for (k=0; k<MAX_STATE; k++) {
     trans[k] = inst[k];
     trans[k].no_transition = 1;
   }
```

Make global constants makros

Result

Removed function pointer



src/shortest-path.c

View

```
@@ -3933,7 +3933,7 @@ void transitions(struct waypoint inst[], struct waypoint trans[])
      struct waypoint *wo=&(inst[c->state_out]);
      if (wo->cost == INF_COST)
        continue;
-     jcost = wo->cost + ss_cost(s);
+     jcost = wo->cost + cost_codesize(s);
      if (jcost <= wi->cost) {
        wi->cost = jcost;
        wi->inst = s;
@@ -3999,7 +3999,7 @@ void optimize_rewrite(PrimNum origs[], int ninsts)
      }
      if (wo->cost == INF_COST)
        continue;
-     jcost = wo->cost + ss_cost(s);
+     jcost = wo->cost + cost_codesize(s);
      if (jcost <= wi->cost) {
        wi->cost = jcost;
        wi->inst = s;
```

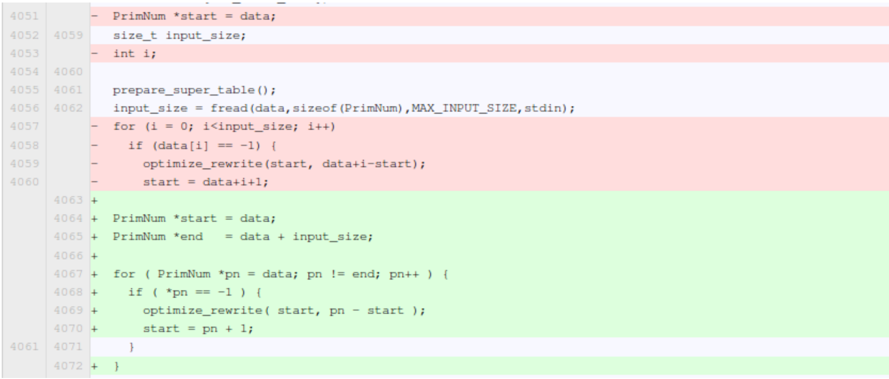
Removed function pointer

Result

Removed is relocateable

Result

Pointer Arithmetik

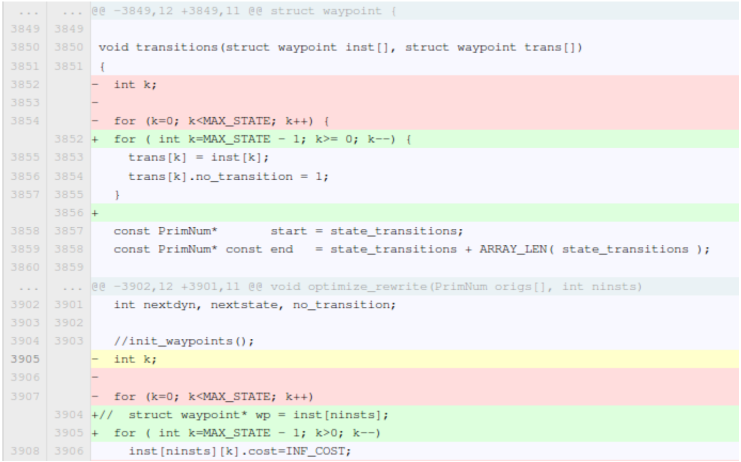


```
-  PrimNum *start = data;
   size_t input_size;
-  int i;

   prepare_super_table();
   input_size = fread(data,sizeof(PrimNum),MAX_INPUT_SIZE,stdin);
-  for (i = 0; i<input_size; i++)
-    if (data[i] == -1) {
-      optimize_rewrite(start, data+i-start);
-      start = data+i+1;
+
+  PrimNum *start = data;
+  PrimNum *end   = data + input_size;
+
+  for ( PrimNum *pn = data; pn != end; pn++ ) {
+    if ( *pn == -1 ) {
+      optimize_rewrite( start, pn - start );
+      start = pn + 1;
     }
+  }
```

Pointer Arithmetik

Result

Loop indices

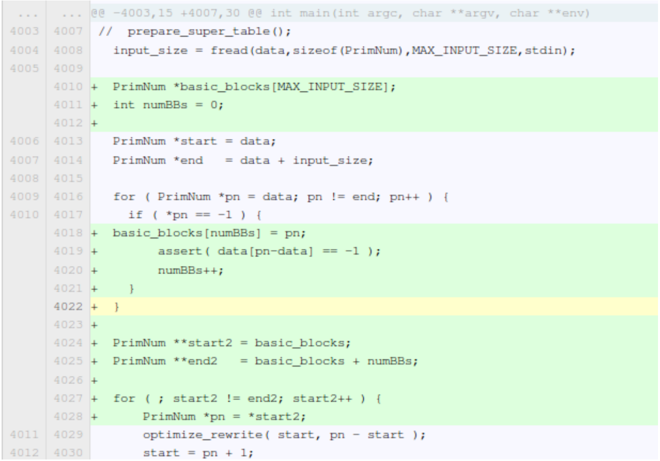


```
@@ -3849,12 +3849,11 @@ struct waypoint {

 void transitions(struct waypoint inst[], struct waypoint trans[])
 {
-  int k;
-
-  for (k=0; k<MAX_STATE; k++) {
+  for ( int k=MAX_STATE - 1; k>= 0; k--) {
     trans[k] = inst[k];
     trans[k].no_transition = 1;
   }
+
   const PrimNum*       start = state_transitions;
   const PrimNum* const end   = state_transitions + ARRAY_LEN( state_transitions );

@@ -3902,12 +3901,11 @@ void optimize_rewrite(PrimNum origs[], int ninsts)
   int nextdyn, nextstate, no_transition;

   //init_waypoints();
-  int k;
-
-  for (k=0; k<MAX_STATE; k++)
+//  struct waypoint* wp = inst[ninsts];
+  for ( int k=MAX_STATE - 1; k>0; k--)
     inst[ninsts][k].cost=INF_COST;
```

Loop indices

Result

Basic blocks

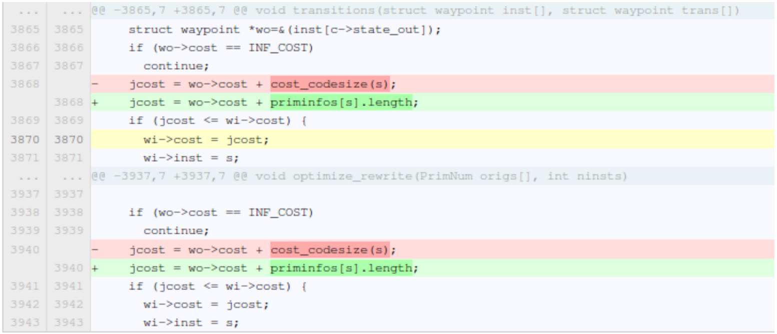


```
@@ -4003,15 +4007,30 @@ int main(int argc, char **argv, char **env)
 //  prepare_super_table();
   input_size = fread(data,sizeof(PrimNum),MAX_INPUT_SIZE,stdin);
 
+  PrimNum *basic_blocks[MAX_INPUT_SIZE];
+  int numBBs = 0;
+
   PrimNum *start = data;
   PrimNum *end   = data + input_size;
 
   for ( PrimNum *pn = data; pn != end; pn++ ) {
     if ( *pn == -1 ) {
+  basic_blocks[numBBs] = pn;
+        assert( data[pn-data] == -1 );
+        numBBs++;
+    }
+  }
+
+  PrimNum **start2 = basic_blocks;
+  PrimNum **end2   = basic_blocks + numBBs;
+
+  for ( ; start2 != end2; start2++ ) {
+      PrimNum *pn = *start2;
       optimize_rewrite( start, pn - start );
       start = pn + 1;
```

Basic blocks

Result

Inline cost codesize



```
@@ -3865,7 +3865,7 @@ void transitions(struct waypoint inst[], struct waypoint trans[])
     struct waypoint *wo=&(inst[c->state_out]);
     if (wo->cost == INF_COST)
       continue;
-    jcost = wo->cost + cost_codesize(s);
+    jcost = wo->cost + priminfos[s].length;
     if (jcost <= wi->cost) {
       wi->cost = jcost;
       wi->inst = s;
@@ -3937,7 +3937,7 @@ void optimize_rewrite(PrimNum origs[], int ninsts)

     if (wo->cost == INF_COST)
       continue;
-    jcost = wo->cost + cost_codesize(s);
+    jcost = wo->cost + priminfos[s].length;
     if (jcost <= wi->cost) {
       wi->cost = jcost;
       wi->inst = s;
```

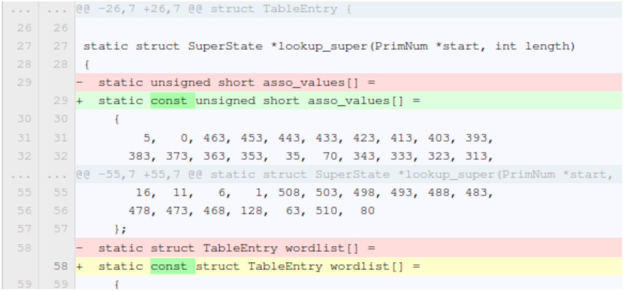
Inline cost codesize

Result

Inlined generated hashfunction into lookup super.

Result

Global arrays to constants

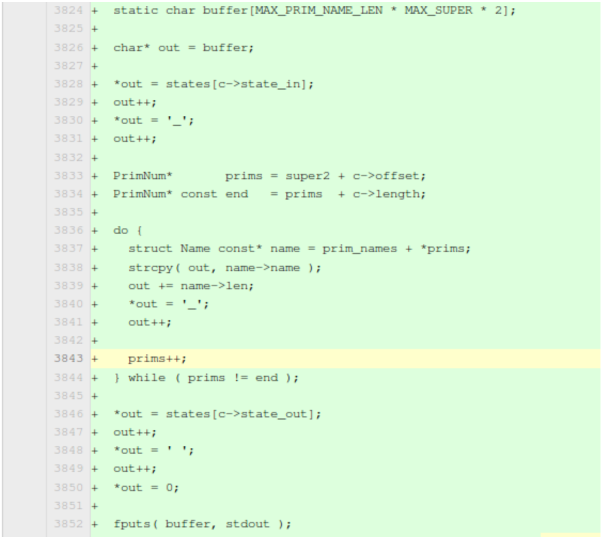


```
@@ -26,7 +26,7 @@ struct TableEntry {

 static struct SuperState *lookup_super(PrimNum *start, int length)
 {
-  static unsigned short asso_values[] =
+  static const unsigned short asso_values[] =
     {
         5,   0, 463, 453, 443, 433, 423, 413, 403, 393,
       383, 373, 363, 353,  35,  70, 343, 333, 323, 313,
@@ -55,7 +55,7 @@ static struct SuperState *lookup_super(PrimNum *start,
        16,  11,   6,   1, 508, 503, 498, 493, 488, 483,
       478, 473, 468, 128,  63, 510,  80
     };
-  static struct TableEntry wordlist[] =
+  static const struct TableEntry wordlist[] =
     {
```

Global arrays to constants

Result

Printinst buffer

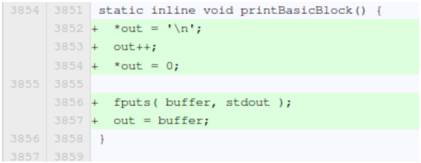


```
+  static char buffer[MAX_PRIM_NAME_LEN * MAX_SUPER * 2];
+
+  char* out = buffer;
+
+  *out = states[c->state_in];
+  out++;
+  *out = '_';
+  out++;
+
+  PrimNum*        prims = super2 + c->offset;
+  PrimNum* const end    = prims  + c->length;
+
+  do {
+    struct Name const* name = prim_names + *prims;
+    strcpy( out, name->name );
+    out += name->len;
+    *out = '_';
+    out++;
+
+    prims++;
+  } while ( prims != end );
+
+  *out = states[c->state_out];
+  out++;
+  *out = ' ';
+  out++;
+  *out = 0;
+
+  fputs( buffer, stdout );
```

Printinst buffer

Result

Printinst basic block into buffer

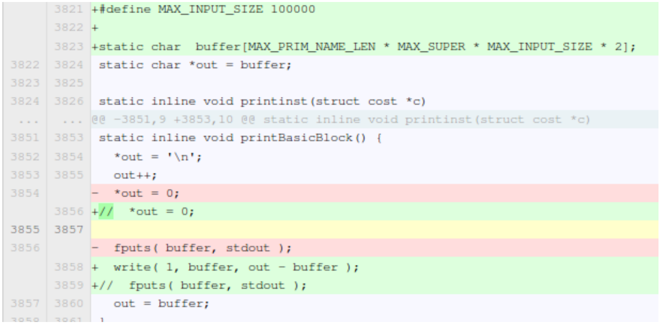


```
 static inline void printBasicBlock() {
+  *out = '\n';
+  out++;
+  *out = 0;

+  fputs( buffer, stdout );
+  out = buffer;
 }

```

Printinst basic block into buffer

Result

Printinst fputs replaced with write

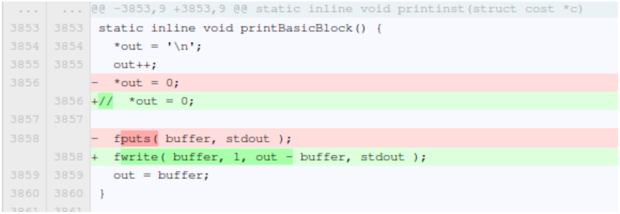


```
+#define MAX_INPUT_SIZE 100000
+
+static char  buffer[MAX_PRIM_NAME_LEN * MAX_SUPER * MAX_INPUT_SIZE * 2];
 static char *out = buffer;

 static inline void printinst(struct cost *c)
@@ -3851,9 +3853,10 @@ static inline void printinst(struct cost *c)
 static inline void printBasicBlock() {
   *out = '\n';
   out++;
-  *out = 0;
+//  *out = 0;

-  fputs( buffer, stdout );
+  write( 1, buffer, out - buffer );
+//  fputs( buffer, stdout );
   out = buffer;
 }
```

Printinst fputs replaced with write

Result

Printinst fputs replaced with fwrite

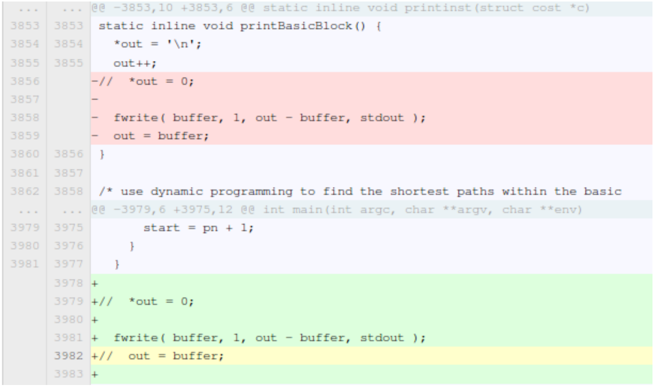


```
@@ -3853,9 +3853,9 @@ static inline void printinst(struct cost *c)
 static inline void printBasicBlock() {
   *out = '\n';
   out++;
-  *out = 0;
+//  *out = 0;

-  fputs( buffer, stdout );
+  fwrite( buffer, 1, out - buffer, stdout );
   out = buffer;
 }
```

Printinst fputs replaced with fwrite

Result

Whole output into buffer



```
@@ -3853,10 +3853,6 @@ static inline void printinst(struct cost *c)
 static inline void printBasicBlock() {
   *out = '\n';
   out++;
-//  *out = 0;
-
-  fwrite( buffer, 1, out - buffer, stdout );
-  out = buffer;
 }
 
 /* use dynamic programming to find the shortest paths within the basic
@@ -3979,6 +3975,12 @@ int main(int argc, char **argv, char **env)
       start = pn + 1;
     }
   }
+
+//  *out = 0;
+
+  fwrite( buffer, 1, out - buffer, stdout );
+//  out = buffer;
+
```

Whole output into buffer

Result

Ultimate optimisation

Result