PEMBANGKIT TEKA-TEKI SILANG DENGAN ALGORITMA *BACKTRACKING* SERTA APLIKASI PERMAINANNYA YANG BERBASIS *WEB*

Hafni Syaeful Sulun dan Rinaldi Munir
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
Jalan Ganesha 10 Bandung 40132
e-mail: hafni.syaeful.sulun@gmail.com, rinaldi-m@stei.itb.ac.id

Abstrak

Teka-teki silang (TTS) merupakan salah satu permainan klasik yang masih populer hingga saat ini dan banyak ditemukan di surat kabar dan buku kumpulan TTS. Walaupun bentuk TTS dari masing-masing penerbit berbeda, ada satu persamaan, yakni bentuk papannya yang memiliki simetri, baik lipat maupun putar, sehingga terlihat menarik dan rapi. Untuk membuatnya secara manual tentu membutuhkan kecermatan karena untuk menentukan satu kata sebagai jawaban harus memperhatikan kata-kata yang terhubung dengannya. Di dalam makalah ini dipaparkan aplikasi pembangkit TTS dan sebuah aplikasi permainan TTS berbasis web yang diberi nama Kata Bersilang. Dengan Kata Bersilang, pembuat TTS (designer) hanya perlu merancang papan TTS, sedangkan jawaban dan pertanyaannya akan dibangkitkan secara otomatis oleh Kata Bersilang dengan algoritma backtracking. Permainan TTS pada Kata Bersilang dibuat mirip dengan permainan TTS yang diterbitkan surat kabar di Indonesia. Penjawab TTS (solver) dapat mengisi TTS yang telah dibuat designer selama masih dalam masa pengisian. Setelah masa tersebut berakhir, akan diumumkan pemenangnya.

Kata kunci: pembangkit TTS, permainan TTS, *backtracking*, Kata Bersilang, *designer*, *solver*.

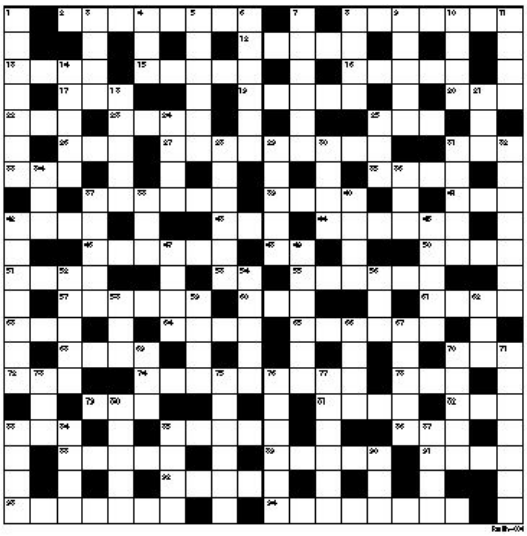
PENDAHULUAN

Teka-teki silang atau biasa disebut TTS merupakan permainan klasik. Selain digunakan untuk menghilangkan rasa jenuh, TTS juga bermanfaat untuk mengasah kemampuan otak dan menambah pengetahuan. Menurut dr. Samino, Sp.S. (K), staf pengajar Departemen Neurologi Fakultas Kedokteran Universitas Indonesia, untuk mencegah penyakit demensia (kepikunan), orang yang berusia paruh baya disarankan untuk melakukan berbagai aktivitas yang merupakan stimulan untuk mencegah turunnya fungsi otak. Salah satu aktivitas tersebut, menurutnya, yaitu mengisi teka-teki silang [2].

Saat ini di Indonesia penerbitan TTS yang umum hanya melalui surat kabar dan buku kumpulan TTS. Dua media inilah yang digunakan masyarakat untuk menyalurkan hobi mengisi TTS. Kedua media ini memiliki perbedaan mendasar. Pada surat kabar pengisi TTS bisa mendapatkan hadiah jika berhasil menjadi pemenang; sedangkan pada buku kumpulan TTS tidak ada pemenang, apalagi hadiah.

Di internet sudah ada situs-situs permainan TTS yang bisa dimainkan secara online, misalnya *New York Times*. Namun, permainan ini hanya bersifat latihan, tidak ada

persaingan antarpengisi untuk memperebutkan hadiah sebagai pemenang. Bahkan, belum ada satu pun permainan TTS *online* yang menggunakan bahasa Indonesia. Karena alasan ini, dengan makalah ini dikembangkan sebuah aplikasi permainan TTS *online* yang berbasis web.



Gambar 1 Contoh Papan TTS dari Kompas [1]

Permainan TTS pada aplikasi yang dibuat ini berlandaskan pada permainan TTS yang diterbitkan sejumlah surat kabar terkemuka di Indonesia. Setiap TTS yang dibuat akan diterbitkan pada waktu tertentu dan bisa diisi oleh pemain sebelum masa pengisiannya berakhir. Setelah itu, pemenang pun bisa didapatkan secara langsung, tidak perlu menunggu, karena penentuan pemenang ditentukan otomatis oleh aplikasi.

Dalam aplikasi ini ada bagian yang cukup krusial, yaitu pembuatan TTS. Untuk membuat sebuah TTS secara manual tentu sangatlah sukar karena harus menyesuaikan antara jawaban yang satu dengan yang lainnya agar terbentuk papan TTS yang utuh. Apalagi agar bentuk TTS memenuhi standar yang telah ditentukan, misalnya bentuknya harus simetris seperti pada Gambar 1.

Tentu aplikasi ini tidak akan membiarkan pembuat TTS membuat TTS sendiri secara manual. Aplikasi ini diharapkan bisa membantu pembuat TTS untuk membuat TTS yang bentuk papannya sesuai dengan standar. Untuk memenuhi persyaratan tersebut, aplikasi ini lebih menekankan pada bentuk papan TTS. Adalah pembuat TTS yang merancang sendiri papan TTS, sedangkan jawaban dan pertanyaan dibangkitkan oleh aplikasi dengan bantuan kamus (data jawaban dan pertanyaan) yang tersimpan dalam basis data. Dengan demikian, diharapkan pembuat TTS hanya perlu memikirkan bagaimana membuat papan TTS yang sesuai dengan standar dan juga menarik.

ALGORITMA *BACKTRACKING*

Teori mengenai algoritma *backtracking* berikut ini mengacu kepada [3]. Algoritma *backtracking* merupakan algoritma yang berbasis pada *Depth First Search* (DFS), tetapi hanya pencarian yang mengarah ke solusi saja yang dipertimbangkan. Artinya, jika dalam pencarian menemui langkah yang tidak mengarah ke solusi, maka akan dicari langkah yang lain.

Pengorganisasian Solusi pada Algoritma *Backtracking*

Semua kemungkinan solusi dari persoalan disebut ruang solusi (*solution space*). Secara formal dapat dinyatakan bahwa jika $x_i \epsilon S_i$, maka

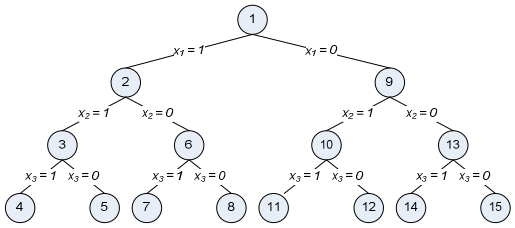
$$S_1 \times S_2 \times ... \times S_n$$

disebut ruang solusi. Jumlah anggota di dalam ruang solusi adalah $|S_1| \cdot |S_2| \cdot ... \cdot |S_n|$.

Sebagai gambaran, tinjau persoalan Knapsack 0/1 untuk $n = 3$. Solusi persoalan dinyatakan sebagai vektor x_1, x_2, x_3 dengan $x_i \epsilon \{0, 1\}$. Ruang solusinya adalah

$$\{0,1\} \times \{0,1\} \times \{0,1\} = \begin{Bmatrix} (0,0,0), (0,1,0), (0,0,1), (1,0,0), \\ (1,1,0), (1,0,1), (0,1,1), (1,1,1) \end{Bmatrix}$$

Dengan kata lain, pada persoalan Knapsack 0/1 dengan $n = 3$ terdapat $2^n = 2^3 = 8$ kemungkinan solusi, yaitu (0, 0, 0), (0, 1, 0), (0, 0, 1), (1, 0, 0), (1, 1, 0), (1, 0, 1), (0, 1, 1), dan (1, 1, 1).



Gambar 2 Ruang solusi persoalan Knapsack 0/1 dengan n = 3

Penyelesaian secara *exhaustive search* adalah dengan menguji setiap kemungkinan solusi. Setiap kemungkinan solusi diperiksa apakah ia memenuhi kendala, yakni menjadi solusi yang layak. Jika ya, maka hitung nilai fungsi objektifnya untuk mencari solusi optimal.

Algoritma *backtracking* memperbaiki pencarian solusi secara *exhaustive search* dengan mencari solusi secara sistematis. Untuk memfasilitasi pencarian ini, maka ruang solusi diorganisasikan ke dalam struktur pohon. Tiap simpul pohon menyatakan status (*state*) persoalan, sedangkan sisi dilabeli dengan nilai-nilai x_i. Lintasan dari akar ke daun menyatakan solusi yang mungkin. Seluruh lintasan dari akar ke daun membentuk ruang solusi.

Pengorganisasian pohon ruang solusi diacu sebagai pohon ruang status (*state space tree*). Tinjau kembali persoalan Knapsack 0/1 untuk n = 3. Ruang solusinya diorganisasikan sebagai pohon ruang status pada Gambar 2. Lintasan dari 1 sampai 4 menyatakan solusi $X = (1, 1, 1)$. Bisa diperhatikan bahwa simpul-simpul diberi urutan nomor sesuai dengan prinsip pencarian DFS. Metode *backtracking* menerapkan DFS dalam pencarian solusi.

Prinsip Pencarian Solusi dengan Metode *Backtracking*

Yang ditinjau di sini adalah pencarian solusi pada pohon ruang status yang dibangun secara dinamis. Langkah-langkahnya adalah sebagai berikut:

1. Solusi dicari dengan membentuk lintasan dari akar ke daun. Aturan pembentukan yang dipakai adalah aturan dalam metode DFS. Simpul-simpul yang sudah dilahirkan dinamakan simpul hidup (*live node*). Simpul hidup yang sedang diperluas dinamakan simpul-E (*expand node*). Simpul dinomori dari atas ke bawah sesuai dengan urutan kelahirannya.

2. Tiap kali simpul-E diperluas, lintasan yang dibangun olehnya bertambah panjang. Jika lintasan yang sedang dibentuk tidak mengarah ke solusi, maka simpul-E tersebut "dibunuh" sehingga menjadi simpul mati (*dead node*). Fungsi yang digunakan untuk membunuh simpul-E adalah dengan menerapkan fungsi pembatas (*bounding function*). Simpul yang sudah mati tidak akan pernah diperluas lagi.
3. Jika pembentukan lintasan berakhir pada simpul mati, maka proses pencarian diteruskan dengan membangkitkan simpul anak yang lainnya. Bila tidak ada lagi simpul anak yang dapat dibangkitkan, maka pencarian solusi dilanjutkan dengan melakukan *backtrack* ke simpul hidup terdekat (simpul orang tua). Selanjutnya simpul ini menjadi simpul-E yang baru. Lintasan baru dibangun kembali sampai lintasan tersebut membentuk solusi.
4. Pencarian dihentikan bila solusi ditemukan atau tidak ada lagi simpul hidup untuk *backtrack*.

Di bawah ini disajikan skema umum algoritma *backtracking* dalam versi rekursif. Algoritma di bawah ini akan menghasilkan semua solusi.

```
procedure RunutBalik(input k: integer)
{ Mencari semua solusi dengan metode
backtracking.
  Masukan: k, yaitu indeks komponen vektor
solusi x[k].
  Keluaran: solusi x = (x[1], x[2], ...,
x[n]).    }
Algoritma:
  for tiap x[k] yang belum dicoba
sedemikian sehingga (x[k] ← T(k)) and
B(x[1], x[2], ..., x[k]) = true do
    if (x[1], x[2], ..., x[k]) adalah
lintasan dari akar ke daun then
      CetakSolusi(x)
    endif
    RunutBalikR(k + 1)
  Endforeach
```

Pemanggilan prosedur pertama kali adalah

ANALISIS

Algoritma *Backtracking* untuk Membangkitkan TTS

Sebelum melakukan pembangkitan jawaban pada papan TTS, ada tahap persiapan yang harus dilalui, yaitu mengubah struktur data yang diterima dari *designer* ke dalam sebuah *array* dan memberi nomor pada papan TTS.

Aplikasi menerima informasi warna kotak-kotak kecil pada papan TTS dari *designer*. Informasi ini berupa variabel bernama grid_x_y di mana x merupakan nomor baris posisi kotak kecil pada papan dan y merupakan nomor kolom posisi kotak kecil pada papan. Nomor baris dan kolom dimulai dari angka 0. Nilai dari variabel ini adalah 0 untuk menunjukkan warna hitam dan 1 untuk menunjukkan warna putih.

Agar lebih mudah dalam pembacaan oleh aplikasi, data papan TTS yang diterima diubah terlebih dulu ke dalam *array*. *Array* ini memiliki dua dimensi untuk menyesuaikan dengan bentuk papan TTS. Semua variabel grid_x_y dibaca satu per satu dan dimasukkan ke dalam variabel grid[x][y] sesuai dengan nomor baris dan nomor kolomnya.

Setelah *array* grid terbentuk, proses selanjutnya adalah melakukan penomoran pada papan TTS. Algoritma paling mudah untuk melakukannya adalah dengan memindai semua kotak baris per baris apakah termasuk kotak yang harus diberi nomor atau tidak. Kotak yang harus diberi nomor adalah kotak putih pertama yang membentuk

sebuah garis kotak putih secara mendatar atau menurun. Untuk garis mendatar, kotak-kotak yang memenuhi aturan tersebut adalah kotak putih yang di sebelah kirinya tidak ada kotak atau kotak hitam dan di sebelah kanannya kotak putih. Untuk garis menurun, kotak-kotak yang memenuhi aturan tersebut adalah kotak putih yang di sebelah atasnya tidak ada kotak atau kotak hitam dan di sebelah bawahnya kotak putih.

Dengan melakukan penomoran ini, aplikasi sekaligus mendapatkan daftar slot untuk jawaban TTS. Variabel yang menyimpan informasi tersebut adalah variabel array_jawaban, yang merupakan sebuah array yang elemennya menyimpan informasi nomor kotak, baris dan kolom kotak, arah penulisan jawaban (mendatar atau menurun), kata/jawaban, petunjuk/pertanyaan, dan kata/jawaban yang telah dicoba selama pencarian solusi. Dalam proses penomoran informasi yang diisi hanya nomor kotak, baris dan kolom kotak, dan arah penulisan jawaban; sedangkan yang lainnya dibiarkan kosong dulu.

Setelah proses penomoran selesai dan daftar kotak yang harus diisi jawaban telah terbuat, proses pembangkitan sudah bisa dilakukan. Pembangkitan ini dilakukan dengan mencoba untuk melengkapi semua elemen dalam daftar jawaban, yakni variabel array_jawaban. Setiap pengambilan kata harus memperhatikan panjang kata dan ketetanggaannya dengan kata yang lain dalam papan. Setiap kata yang dicoba pada satu slot jawaban harus diingat agar kata yang sama tidak diambil dua kali. Selain itu, kata yang diambil juga tidak boleh sudah ada dalam daftar jawaban.

Pseudo-code algoritma *backtracking* yang digunakan untuk membangkitkan teka-teki silang adalah sebagai berikut:

```
Kamus:
array_jawaban: array of jawaban { jawaban merupakan array dengan
indeks no, baris, kolom, arah, kata, petunjuk, kata_dicoba }
grid_huruf: array of char { representasi papan TTS }

function generate()
{ Membangkitkan jawaban pada papan TTS dengan mencoba untuk
mengisi semua array_jawaban dan grid_huruf. Mengembalikan TRUE
jika semua jawaban berhasil dibangkitkan, FALSE jika tidak
berhasil. }
Kamus:
  solusi_ditemukan: boolean
  indeks_jawaban: integer
  kata: string
Algoritma:
  solusi_ditemukan <- TRUE
  indeks_jawaban <- 0

  while (solusi_ditemukan and indeks_jawaban < jumlah elemen
  array_jawaban) do
    kata <- ambil kata yang memenuhi syarat
    if (kata tidak ditemukan) then
      if (indeks_jawaban = 0) then
        solusi_ditemukan <- FALSE
      else
        { backtrack }
        while (huruf-huruf pada kata sekarang belum ada yang
         terhapus) do
          hapus huruf-huruf pada kata sebelumnya di grid_huruf
           yang tidak berbagi pakai dengan kata yang lain
         indeks_jawaban<-indeks_jawaban - 1
        endwhile
      endif
    else
array_jawaban[indeks_jawaban]['kata'] <- kata
array_jawaban[indeks_jawaban]['kata_dicoba'][] <- kata
      masukkan setiap huruf dari kata ke dalam papan
(grid_huruf)
      indeks_jawaban <- indeks_jawaban + 1
    endif
```

```
  endwhile

  ← solusi_ditemukan
```

Struktur Data TTS

Ukuran papan TTS yang tidak tentu mengharuskan struktur data TTS yang fleksibel. Jika dibuat kolom sebanyak jumlah kolom maksimal yang mungkin dalam papan TTS, maka tidaklah efisien. Jika ini diterapkan, maka akan ada banyak kolom yang tidak bernilai jika ukuran papan TTS minimal.

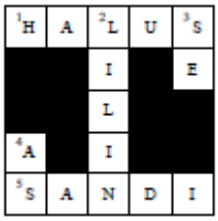
Begitu pula jika tabel jawaban hanya memiliki satu kolom jawaban sehingga satu TTS memiliki banyak baris dalam tabel jawaban tersebut. Dengan diterapkannya tabel seperti ini maka proses pengambilan data akan memakan waktu cukup lama karena ada penyaringan data yang sangat banyak khususnya di tabel jawaban. Selain itu, untuk menyimpan sebuah TTS aplikasi juga harus melakukan pengisian data yang banyak karena banyaknya jawaban.

Berdasarkan alasan-alasan tersebut, dibuatlah sebuah struktur yang lebih efisien. Struktur ini hanya menggunakan sebuah tabel untuk menyimpan informasi TTS lengkap dengan jawabannya. Ilustrasi tabelnya dapat dilihat pada Tabel 1.

Tabel 1 Tabel TTS setelah diringkas

TTS			
ID	**Nama**	**Pembuat**	**Papan**

Kolom yang menyimpan data papan TTS adalah kolom papan. Kolom ini diisi dengan informasi papan TTS, yaitu jumlah baris, jumlah kolom, dan daftar jawaban. Agar mudah dalam pembacaan dan penulisan, data ini ditulis dalam format JSON. Untuk lebih jelasnya, berikut ini diberikan sebuah contoh.
Papan TTS yang akan disimpan dapat dilihat pada Gambar 3.



Gambar 3 Contoh papan TTS yang akan disimpan ke dalam basis data

Data dalam format JSON dari papan TTS di atas adalah sebagai berikut:

```
{"jumlah_baris":5,"jumlah_kolom":5,"array_jawaban":[{"no":1,"baris":0,"kolom":0,"arah":"mendatar","jawaban":"HALUS","pertanyaan":"Tidak
kasar"},{"no":2,"baris":0,"kolom":2,"arah":"menurun","jawaban":"LILIN","pertanyaan":"Pengganti lampu"
},{"no":3,"baris":0,"kolom":4,"arah":"menurun","jawaban":"SE","pertanyaan":"Sarjana
Ekonomi"},{"no":4,"baris":3,"kolom":0,"arah":"menurun","jawaban":"AS","pertanyaan":"Amerika
Serikat"},{"no":5,"baris":4,"kolom":0,"arah":"mendatar","jawaban":"SANDI","pertanyaan":"Kode, rahasia"}]}
```

Untuk menampilkan papan TTS dari data ini cukup mudah. Pertama, dibuatlah papan TTS berukuran sesuai dengan jumlah baris dan jumlah kolom dari data. Set semua kotak kecil berwarna hitam. Dengan membaca jawaban satu per satu, letakkan jawaban pada papan sesuai dengan nomor, posisi, dan arahnya. Jika semua jawaban sudah diletakkan, maka papan TTS sudah jadi.

Basis Data Jawaban TTS

Faktor kunci keberhasilan pembangkitan TTS terletak pada kumpulan jawaban yang tersedia. Semakin banyak jawaban tersebut, semakin besar kemungkinan berhasilnya TTS terbangkitkan. Oleh karena itu, dibutuhkan teknik pemrosesan data sedemikian sehingga waktu yang diperlukan sekecil mungkin walaupun ukuran basis data sangat besar.

Demi memenuhi salah satu kebutuhan TTS yang dibuat, yakni tidak terbatasnya jumlah huruf dalam setiap jawaban, tabel yang digunakan untuk menyimpan jawaban TTS hanya satu buah. Hal ini tentu akan mengorbankan waktu yang diperlukan ketika aplikasi membutuhkan data jawaban dengan jumlah huruf tertentu karena akan dilakukan penyaringan data yang sangat banyak. Ilustrasi tabel untuk menyimpan data jawaban TTS tersebut dapat dilihat pada Tabel 2.

Untuk mengatasi masalah bahwa aplikasi membutuhkan data jawaban yang banyak setiap kali akan membangkitkan sebuah TTS, dibuatlah sebuah teknik pengambilan data yang diterapkan di sisi aplikasi. Daripada mengambil data langsung ke basis data dengan melakukan *query* setiap kali aplikasi membutuhkan sebuah jawaban, lebih baik aplikasi mengambil banyak data jawaban sekaligus dalam satu kali *query* kemudian disimpan dalam variabel.

Teknik ini mirip dengan teknik *caching*, yakni aplikasi akan memeriksa apakah ada jawaban di memori aplikasi yang masih bisa digunakan ketika aplikasi membutuhkannya. Jika ada, maka aplikasi menggunakan jawaban tersebut. Jika tidak ada, maka aplikasi akan melakukan *query* ke basis data untuk mengambil lagi data jawaban yang dibutuhkan. Tabel untuk menyimpan data jawaban TTS dapat dilihat pada Tabel 2.

Tabel 2 Tabel jawaban TTS

Jawaban		
ID	**Jawaban**	**Pertanyaan**

HASIL DAN PENGUJIAN

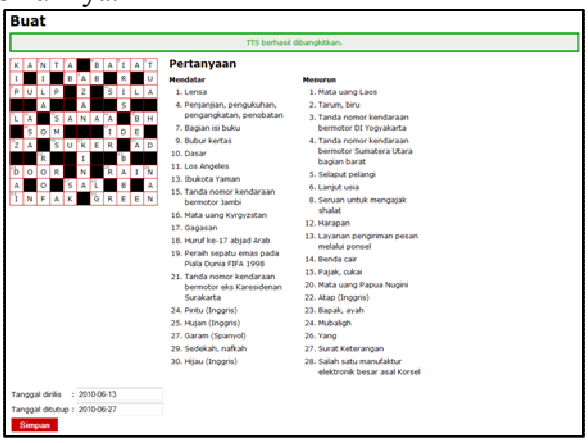
Aplikasi permainan TTS yang dibuat diberi nama Kata Bersilang. Aplikasi tersebut dibuat berbasiskan web serta ditulis dalam PHP dengan *framework* CodeIgniter dan pustaka Javascript jQuery dan jQuery UI.

Kata Bersilang dibuat dengan empat fitur utama untuk dua tipe pengguna. Pengguna bertipe *designer* mempunyai hak untuk membuat TTS dan melihat peringkat penjawab TTS, sedangkan pengguna bertipe *solver* mempunyai hak untuk mengisi TTS dan melihat pemenang TTS.

Berikut ini penjelasan singkat mengenai keempat fitur dalam Kata Bersilang.

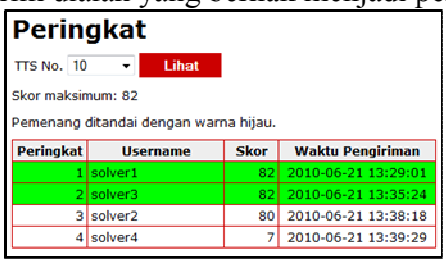
1. Membuat TTS

Untuk membuat sebuah TTS, *designer* mula-mula menentukan ukuran papan TTS yang akan dibuat. Selanjutnya aplikasi akan menampilkan halaman untuk merancang papan TTS. Setelah *designer* selesai merancang papan TTS tersebut dan mengirimnya ke aplikasi, proses pembangkitan jawaban dan pertanyaan dilakukan aplikasi dan hasilnya akan ditampilkan kemudian.
Gambar 4 menunjukkan tampilan setelah aplikasi berhasil membangkitkan jawaban dan pertanyaan untuk papan TTS yang dibuat *designer*.
Jika pembangkitan berhasil, maka *designer* bisa menentukan masa pengisian TTS dan mempublikasikannya.



Gambar 4 *Screenshot* membuat TTS

2. Melihat peringkat pengisi TTS
Designer bisa melihat peringkat pengisi TTS yang dipublikasikannya setiap saat. Yang bisa menjadi pemenang TTS adalah pengisi TTS yang berhasil menjawab dengan benar semua pertanyaan pada papan TTS. Pemenang dibatasi hanya tiga orang. Jika ada lebih dari tiga orang yang berhasil menjawab dengan benar penentuan pemenang ditentukan berdasarkan waktu pengiriman jawaban. Siapa yang lebih dulu mengirim dialah yang berhak menjadi pemenang.

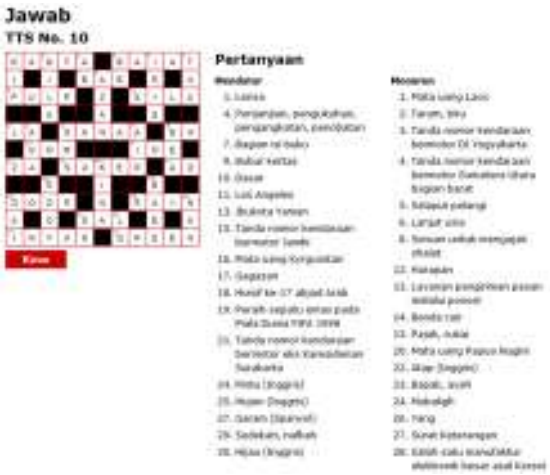


Peringkat	Username	Skor	Waktu Pengiriman
1	solver1	82	2010-06-21 13:29:01
2	solver3	82	2010-06-21 13:35:24
3	solver2	80	2010-06-21 13:38:18
4	solver4	7	2010-06-21 13:39:29

Gambar 5 *Screenshot* melihat peringkat pengisi TTS

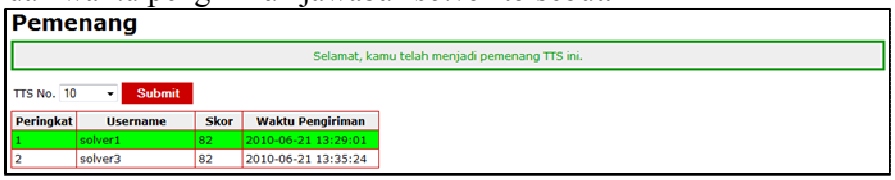
Agar lebih mudah dibaca, pemenang TTS ditandai dengan warna hijau. Tampilan peringkat tersebut dapat dilihat pada Gambar 5.

3. Mengisi TTS
Solver bisa mengisi TTS yang masih dalam masa pengisian. Sebelum mengisinya, *solver* memilih lebih dulu TTS mana yang akan diisi. Tampilan pengisian TTS dapat dilihat pada Gambar 6.



Gambar 6 *Screenshot* mengisi TTS

4. Melihat pemenang TTS
   Setelah masa pengisian berakhir, pemenang TTS dapat ditentukan. *Solver* pun bisa melihat siapa saja yang menjadi pemenang. Selain itu, juga ditampilkan peringkat, skor, dan waktu pengiriman jawaban *solver* tersebut.



Gambar 7 *Screenshot* melihat pemenang TTS

Penjelasan beserta *screenshot* di atas sekaligus sebagai hasil pengujian fitur yang ada dalam Kata Bersilang. Khusus untuk fitur pembangkitan TTS, berikut ini ditampilkan TTS ukuran terbesar yang diuji dan berhasil dibangkitkan, yakni ukuran 15 × 15.

Gambar 8 Hasil pembangkitan TTS ukuran 15 × 15

Berdasarkan pengujian yang telah dilakukan, dapat diketahui bahwa aplikasi permainan TTS yang dibuat mampu berjalan dengan baik. Terkait fitur pembuatan TTS; semakin besar ukuran papan TTS yang dibuat, perangkat lunak semakin kesulitan membangkitkannya sehingga waktu yang diperlukan cenderung semakin lama. Hal ini disebabkan oleh semakin banyaknya kendala keterhubungan antara satu kata dengan kata yang lain jika ukuran papan TTS semakin besar.

KESIMPULAN

Berdasarkan hasil pengujian didapatkan kesimpulan sebagai berikut:

1. Algoritma *backtracking* dapat diterapkan untuk membangkitkan papan TTS kosong menjadi papan TTS yang terisi penuh dengan jawaban dan pertanyaan.
2. Pembangkitan TTS dengan algoritma *backtracking* telah berhasil diimplementasikan ke dalam sebuah perangkat lunak berbasis web. Perangkat lunak tersebut diberi nama Kata Bersilang.
3. Ukuran terbesar papan TTS yang dapat dibangkitkan Kata Bersilang dengan waktu relatif cepat (tidak lebih dari 5 menit) adalah 15 × 15.
4. TTS yang telah berhasil dibangkitkan dan dirilis, bisa diisi oleh pengguna bertipe *solver* selama masih dalam masa pengisian TTS yang bersangkutan.
5. Setelah masa pengisian TTS berakhir, pembuat TTS bisa melihat peringkat akhir penjawab dan *solver* bisa melihat daftar pemenang.

REFERENSI

[1] Harian Kompas. 2002. *Teka-Teki Silang Kompas Minggu No. 1150 – Minggu, 27 Januari 2002*. http://www2.kompas.com/kompas-cetak/0201/27/hiburan/teka23.htm. Diakses pada tanggal 2 Februari 2009 WIB.

[2] Kompas Group. 2006. *Asah Otak, Cegah Pikun*! http://www2.kompas.com/ver1/Kesehatan/0709/27/112851.htm. Diakses pada tanggal 2 Februari 2009 WIB.

[3] Munir, Rinaldi. 2007. *Diktat Kuliah IF2251 Strategi Algoritmik*. Bandung: Program Studi Teknik Informatika STEI ITB.